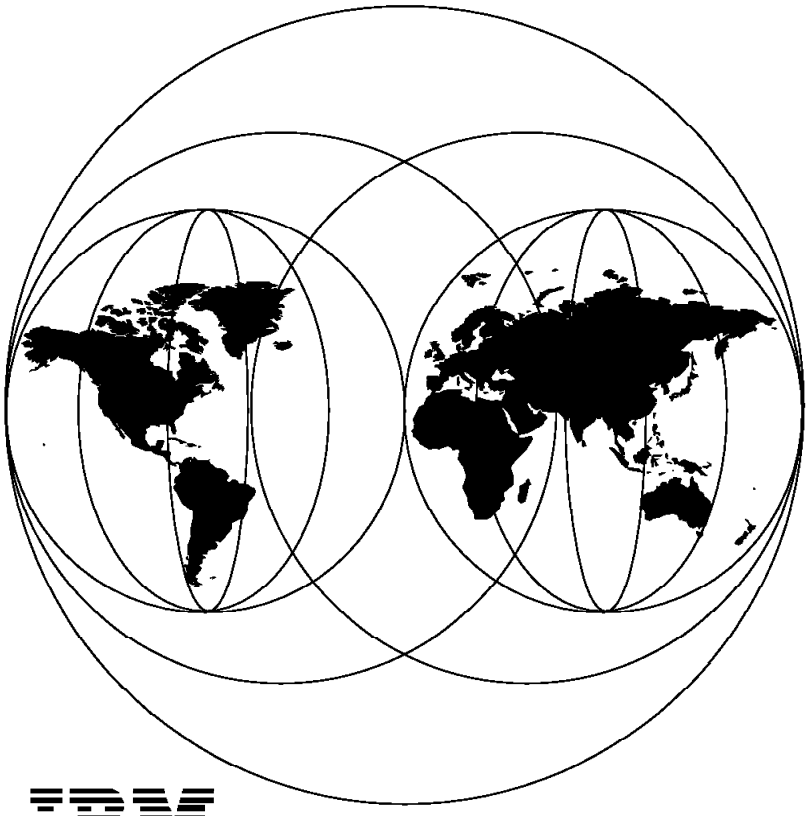
SG24-4867-00

TME 10 Cookbook for AIX
Systems Management and Networking Applications

December 1996



IBM

International Technical Support Organization
Raleigh Center

IBM

International Technical Support Organization

SG24-4867-00

TME 10 Cookbook for AIX
Systems Management and Networking Applications

December 1996

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 537.

First Edition (December 1996)

This edition applies to TME 3.0 for use with the AIX Operating System.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1996. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Figures ix

Tables xix

Preface xxi
How This Redbook Is Organized xxi
The Team That Wrote This Redbook xxiii
Comments Welcome xxiii

Part 1. Introduction, Installation and Base Platform Capabilities 1

Chapter 1. Introduction to the Tivoli Management Environment 3
1.1 Components of the Tivoli Management Platform 4
1.1.1 TME Resources 5
1.1.2 Tivoli Management Region (TMR) 5
1.1.3 Policy Regions 5
1.1.4 Policies 6
1.1.5 Profile Endpoints 6
1.1.6 Managed Nodes and PC Managed Nodes 6
1.1.7 Profiles 6
1.1.8 Profile Managers 7
1.2 Tivoli Administrators 7
1.2.1 Authorization Roles 8
1.2.2 Notice Groups 9
1.3 A Practical Example of a TME Management Concept 10

Chapter 2. Planning and Installing the Tivoli Management Platform 17
2.1 TME Management Concept 17
2.2 The Environment at the ITSO 20
2.3 Security 20
2.3.1 Encryption Levels and Passwords 20
2.4 Installing the TME Management Platform 23
2.4.1 Planning the Installation 23
2.4.2 Installing TME on a Server 25
2.4.3 How to Start the Tivoli Management Environment 28
2.4.4 Installing UNIX Managed Nodes as TME Clients 29
2.4.5 Installing PC Managed Node as TME Client 31
2.4.6 TME Service Pack Installation 40
2.5 Product Installation 40
2.5.1 Product Installation Using the TME Desktop 41
2.6 Diagnosing Installation Problems 44
2.6.1 UNIX Environment Problems 44
2.6.2 Space Problems 45
2.6.3 Reinstallation 49
2.6.4 Problems Installing Products and Patches 49
2.6.5 TME Client Installation Problems 50
2.6.6 Some Useful Diagnostic Commands 52
2.7 Setting up a Backup Schedule 52
2.7.1 Backup Prerequisites 53
2.7.2 Example Backup Schedule 53

© Copyright IBM Corp. 1996

Chapter 3. Configuring the TME Management Platform . . . 59
3.1.1 Connecting Tivoli Management Regions . . . 59
3.1.2 Create Policy Regions . . . 61
3.1.3 Arrange Policy Regions . . . 62
3.1.4 TMR Resource Updates . . . 64
3.1.5 Create Profile Managers . . . 68
3.1.6 Subscribe to Profile Managers . . . 69
3.1.7 Create Administrators . . . 70
3.2 Some Performance Considerations within the TME Environment . . . 76
3.2.1 Installation of the Performance Toolkit . . . 77
3.2.2 Approach . . . 77
3.2.3 Analyzing the Data . . . 79
3.2.4 Results of Data Analysis . . . 80
3.2.5 Observations . . . 80

Chapter 4. Task Libraries, Tasks and Jobs . . . 81
4.1 A Simple Task Library Example . . . 81
4.1.1 Create Task Libraries . . . 81
4.1.2 Create Tasks . . . 82
4.1.3 Create Jobs . . . 84
4.2 A More Complex Task Example . . . 85
4.2.1 Create an Administrator . . . 85
4.2.2 Create a Task Library . . . 90
4.2.3 Create a Task . . . 91
4.2.4 Create Task Libraries and Tasks from the Command Line . . . 92
4.2.5 Executing the Task . . . 93
4.2.6 Administrator Roles Needed to Execute a Task . . . 97
4.2.7 Executing a Task with Arguments from the Command Line . . . 98
4.2.8 Executing a Task with Arguments from the Desktop . . . 99

Part 2. Deployment Applications . . . 101

Chapter 5. Tivoli/Courier . . . 103
5.1 Courier Features . . . 103
5.1.1 File Packages . . . 103
5.1.2 Nested File Packages . . . 104
5.1.3 File Package Operations . . . 105
5.1.4 Configuration Programs . . . 105
5.1.5 Import/Export File Packages . . . 106
5.1.6 WAN-Smart Capabilities . . . 107
5.1.7 Security . . . 107
5.1.8 Resources . . . 108
5.2 Installation . . . 109
5.2.1 Planning the Installation . . . 109
5.3 Configuring Courier Distribution Environment . . . 110
5.3.1 Policy Regions . . . 110
5.3.2 Network Distribution Environment . . . 110
5.3.3 Create Profile Managers . . . 113
5.4 Packaging and Distributing Software . . . 119
5.4.1 Create the INed File Package . . . 120
5.4.2 Software Distribution of INed . . . 133
5.4.3 Create the MQSeries for AIX File Package . . . 149
5.4.4 Software Distribution of MQSeries for AIX . . . 159
5.4.5 Create the MQSeries for NT File Package . . . 173

5.4.6 Software Distribution of MQSeries for NT . . . 185
5.5 Problems Encountered . . . 202
5.5.1 During Installation of Courier . . . 202
5.5.2 Trying to Move a Managed Node from One TMR to Another . . . 203

Chapter 6. Tivoli/Inventory . . . 205
6.1 Tivoli/Inventory Components . . . 206
6.1.1 Tivoli/Inventory Server Component . . . 206
6.1.2 PC Inventory Scanning Component . . . 206
6.1.3 UNIX Inventory Scanning Component . . . 206
6.1.4 Control File Data Store Component . . . 206
6.1.5 TME Configuration Repository Component . . . 207
6.2 Installation . . . 207
6.2.1 Installing the RDBMS . . . 207
6.2.2 Installing Tivoli/Inventory . . . 209
6.2.3 Creating the Inventory User Account on the RDBMS . . . 213
6.2.4 Installing Configuration Repository Database . . . 214
6.2.5 Installing the PC Managed Nodes . . . 214
6.2.6 Installing Managed Nodes . . . 216
6.3 Using Tivoli/Inventory . . . 217
6.3.1 Set Resource Roles for Inventory . . . 217
6.3.2 Set Up an Inventory Profile . . . 218
6.3.3 Scanning the Environment . . . 223
6.3.4 Viewing the Scanned Data . . . 231
6.4 Information without Using Tivoli/Inventory . . . 234
6.5 TME Query Facility . . . 235
6.5.1 Create Query Libraries . . . 236
6.5.2 Create Queries . . . 237
6.5.3 Extend the Query Facility . . . 238
6.6 TME Configuration Repository . . . 241
6.6.1 TME Configuration Repository Data Model . . . 241
6.6.2 Enterprise Specific Configuration Data . . . 242
6.7 Create a Reference Model . . . 242
6.7.1 Add a Software Reference Model . . . 242
6.7.2 Assign a Reference Model to a System . . . 244
6.7.3 Compound Reference Models . . . 244
6.7.4 Change Notification . . . 245

Chapter 7. Tivoli/Courier and Tivoli/Inventory Interoperability . . . 249
7.1 Create Tivoli/Inventory Query . . . 249
7.2 Use Query in Software Distributions . . . 249

Chapter 8. TME 10 User Administration . . . 253
8.1 Planning, Installation and Initial Configuration . . . 253
8.1.1 Installing Tivoli/Admin . . . 254
8.2 User ID Management Example . . . 254
8.2.1 Setting the Scene . . . 255
8.2.2 Creating User Profiles . . . 255
8.2.3 User Profile Policies . . . 258
8.2.4 Managing User Profiles . . . 261
8.2.5 Setting Subscribers . . . 267
8.2.6 Create a New User . . . 271
8.2.7 Distribute User Profiles . . . 276
8.2.8 Modifying User Definitions . . . 285
8.2.9 Deleting User Records . . . 286

8.2.10 How Tivoli Admin Updates System Files 287
8.2.11 Invoking Tivoli/Admin Functions from the Command Line 288

Part 3. System Monitoring and Event Handling Applications 291

Chapter 9. TME 10 Distributed Monitoring 293
9.1.1 Installing the Sentry Application 294
9.2 Examples of Using the Sentry Monitors for a Web Server 295
9.2.1 Web Server Configuration 295
9.2.2 Configuring the Sentry Monitors 296
9.2.3 Creating the Profile Manager 296
9.2.4 Monitor Profiles 298
9.2.5 Monitors 299
9.2.6 Monitoring Using the Asynchronous String Interface 304
9.2.7 Creating an Indicator Collection 310
9.2.8 Preparing to Distribute the Profiles 312
9.2.9 Profile Distribution 313
9.3 Planning the Sentry Monitors 314

Chapter 10. Introduction to the TME 10 Enterprise Console 319
10.1 Planning the Installation 320
10.2 T/EC Installation 321
10.3 Configuring the T/EC 323
10.3.1 Defining the Administrators 324
10.3.2 Initial Configuration of the T/EC Rulebase 329
10.3.3 Importing Event Classes 332
10.3.4 Defining T/EC Groups and Sources 336
10.3.5 Configuring T/EC Server Parameters 340
10.4 Using the Tivoli Enterprise Console Display 341
10.4.1 Tuning the T/EC Interface 343
10.4.2 T/EC Tasks 344
10.4.3 Running a Task Against a Particular Event 346
10.4.4 Adding an Automated Task 347
10.4.5 Creating a New T/EC Task 349
10.5 Useful Commands 353

Chapter 11. Tivoli/Enterprise Console Adapters 355
11.1 Installing the Event Adapters 355
11.2 Adapter Configuration Files 356
11.3 The Logfile Adapter 356
11.3.1 Installation of the T/EC Logfile Adapter 357
11.3.2 Configuration of the Logfile Adapter 361
11.4 NetView for AIX and Openview Adapters 367
11.4.1 Installation and Initial Configuration 368
11.5 The NetView for AIX Ruleset Event Adapter 370
11.5.1 Installation and Initial Configuration 371
11.5.2 Configuring the NetView Rulesets 373
11.6 The SNMP Adapter 375
11.7 The Tivoli Sentry Event Adapter 376
11.7.1 Importing Sentry Event Classes 376
11.7.2 A Sentry Monitor Example 376
11.8 Windows NT Adapter 379
11.8.1 Installing the NT Adapter 379
11.8.2 Configuring T/EC for the NT Logfile Adapter 380

11.8.3 An Example of NT Logfile Adapter Events . . . 380
11.9 Generating T/EC Events from the Command Line . . . 381

Chapter 12. More Advanced TME 10 Enterprise Console Customization . . . 383
12.1 Creating Enterprise Console Rulesets . . . 383
12.1.1 An Introduction to T/EC Rules . . . 384
12.1.2 Using the Rule Builder . . . 384
12.1.3 Coding Event Rules Manually . . . 389
12.1.4 Tracing Rules . . . 391
12.2 Extending the Logfile Adapter to Manage a Distributed Application . . . 393
12.2.1 Test Scenario . . . 393
12.2.2 Defining New Logfile Event Classes . . . 393
12.2.3 Mapping Message Formats to Event Classes . . . 396
12.2.4 Updating the Logfile Adapter Configuration Files . . . 397
12.2.5 Debugging the Logfile Adapter . . . 398
12.2.6 Preparing Event Consoles to Receive MQS Events . . . 399
12.2.7 Testing Events from MQ Manager on the T/EC . . . 400
12.2.8 Create a Task to Restart the Channel . . . 402
12.2.9 Executing a TME Task from a T/EC Ruleset . . . 409
12.2.10 Results of MQ Series Message Automation . . . 411
12.3 Extending the NetView for AIX/OpenView Adapters . . . 413
12.3.1 Systems Monitor MLM Integration . . . 414
12.3.2 Managing SNMP Network Devices Using the NetView Ruleset Adapter . . . 420

Part 4. Integrating Management Applications into TME . . . 425

Chapter 13. Tivoli/Plus Modules . . . 427
13.1 Tivoli/Plus Modules in Practice . . . 427
13.1.1 The Remedy Action Request System . . . 428
13.1.2 Installation . . . 428
13.1.3 Using the ARS /Plus Module T/EC Integration Feature . . . 432
13.2 ADSM Tivoli/Plus Module . . . 435
13.2.1 Installation . . . 435
13.2.2 Using the ADSM /Plus Module to Install ADSM Backup Clients . . . 436
13.2.3 ADSM Log Interface with the TME 10 Enterprise Console . . . 439

Chapter 14. TME 10 Net.Commander . . . 441
14.1 Installing Net.Commander . . . 441
14.2 Customizing and Managing a Web Server . . . 442
14.3 Using Net.Commander Functions . . . 445
14.3.1 Tasks and Jobs . . . 446
14.3.2 Monitors for the Web Server . . . 446
14.3.3 Getting T/EC Events from the Web Server . . . 447
14.4 Other Internet Servers . . . 450
14.5 Firewalls . . . 451
14.6 An Example of Extending the Log File Adapter to Monitor a Firewall . . . 451
14.6.1 Preparation of the Firewall . . . 452
14.6.2 Customizing the Log File Adapter for the Firewall . . . 452

Chapter 15. Adding Function to the TME Desktop . . . 455
15.1 Using the Task Library Language (TLL) . . . 455
15.1.1 Getting Started . . . 455
15.1.2 Create a Task Library Using TLL . . . 457

15.1.3 Executing a Task with Arguments from the Desktop 461
15.1.4 Create a Job to Execute Set_isTME_NODE 464
15.2 Configure a TME Policy Object for Task Library 467
15.2.2 Validation Policy 475
15.3 Using the Tivoli Application Extension Facility (AEF) 479

Chapter 16. Integrating NetView for AIX with Other TME Functions 495
16.1 Customizing the NetView for AIX Environment 495
16.2 Configuring NetView for AIX Rulesets 497
16.2.1 Defining Ruleset Working with Tivoli 500
16.3 NetView for AIX Menu Integration 502
16.4 Creating NetView for AIX Collections from the Tivoli Database 510
16.4.1 Collection Definitions 511
16.4.2 Creating the Policy Region Collections 511

Appendix A. TME Configuration Repository 519
A.1 Example Data via the INVENTORYDATA View 519
A.2 TME Configuration Repository Views 522
A.3 Configuration Repository Tables 524

Appendix B. Special Notices 537

Appendix C. Related Publications 541
C.1 International Technical Support Organization Publications 541
C.2 Redbooks on CD-ROMs 541
C.3 Other Publications 541
C.4 Other CD-ROMs 542
C.5 Other Diskettes 542

How To Get ITSO Redbooks 543
How IBM Employees Can Get ITSO Redbooks 543
How Customers Can Get ITSO Redbooks 544
IBM Redbook Order Form 545

List of Abbreviations 547

Index 549

Figures

1. TME Architecture . . . 3
2. Example of a TMR Structure . . . 11
3. Example of a Policy Region Structure . . . 12
4. Example of a Profile Manager Structure . . . 14
5. Structure by Project . . . 18
6. Policy Regions . . . 19
7. Installation Directories . . . 26
8. Installation Specific Information . . . 27
9. Initial Desktop . . . 28
10. Set Managed Resources . . . 29
11. Create Managed Node . . . 30
12. Add Client Dialog . . . 31
13. Devices Defined at Your PC . . . 32
14. Files and Directories on the CD-ROM . . . 32
15. Directory of TCP/IP Client Software . . . 33
16. Welcome Panel . . . 33
17. Platforms . . . 34
18. Confirm Continuing Installation . . . 34
19. Enter Destination Drive . . . 35
20. Enter Destination Directory . . . 35
21. Accept Installation Settings . . . 36
22. Select Startup Functions . . . 36
23. Confirm Continuing Installation . . . 37
24. Enter Name of the Machine . . . 37
25. Remote Server for IP Synchronization . . . 38
26. Remote Server Name . . . 38
27. Allow IP Synchronization at Bootup . . . 38
28. Enter Update Interval . . . 39
29. Installation Completed . . . 39
30. Patch Installation . . . 40
31. Install Product Dialog . . . 41
32. File Browser Dialog . . . 42
33. Product Install Dialog . . . 43
34. Product Install Dialog at the End of Installation . . . 44
35. Not Enough Space to Install Message . . . 45
36. Continuing without Enough File System Space . . . 46
37. Insufficient Space for the Server Database . . . 47
38. Seemingly Successful Installation Messages . . . 48
39. Messages When Installing with Some Files Already Installed . . . 49
40. Warning on First Product Install . . . 50
41. Error Messages from Client Install . . . 51
42. Backup Tivoli Management Region Dialog . . . 54
43. Select Notice Groups Panel . . . 54
44. Set Retry/Cancel Options Panel . . . 55
45. Add Scheduled Job Panel . . . 56
46. Browse Scheduled Jobs Panel . . . 57
47. Connect to a Remote TMR . . . 60
48. Confirm TMR Connection . . . 61
49. Create Policy Region . . . 62
50. Top Level Policy Regions . . . 63
51. Subregions of Policy Region Top . . . 64

© Copyright IBM Corp. 1996

52. Update Resources from Multiple TMRs . . . 66
53. Add Scheduled Job . . . 68
54. Create Profile Manager . . . 69
55. Add Subscribers to a Profile Manager . . . 70
56. Create Administrator Dialog . . . 72
57. Set TMR Roles . . . 73
58. Set Resource Roles . . . 74
59. Set Login Names . . . 74
60. Set Notice Groups . . . 75
61. Create Tivoli Administrator from Command Line . . . 76
62. Performance Data Capture Values . . . 78
63. Snapshot of Data Collected . . . 79
64. Task Library . . . 82
65. Create Task . . . 83
66. Create Job . . . 84
67. Create a New Administrator . . . 86
68. Set TMR Roles . . . 87
69. Set Resource Roles . . . 87
70. Set Logins . . . 88
71. New Administrator Icon Appears . . . 88
72. Open Administrator Desktop . . . 89
73. Lynn's Desktop with One Policy Region . . . 89
74. Insufficient Authorization to Create a Task Library . . . 90
75. Creating a New Task Library . . . 91
76. Creating a New Task Library . . . 92
77. Executing a Task . . . 94
78. Execute Task Dialog . . . 95
79. Task Fails to Execute Due to Insufficient Authorization . . . 95
80. Modifying the Policy Region Resource Roles . . . 96
81. Successful Task Execution . . . 97
82. Result of the Show_Filesystems Task with Arguments Supplied . . . 99
83. Tivoli/Courier Resources . . . 109
84. Software Distribution Scenario . . . 111
85. Create Profile Manager AIXMachs . . . 113
86. Policy Region SoftDist . . . 114
87. Subscribers to AIXMachs . . . 114
88. Profile Manager AIXMachs . . . 115
89. Create File Package INed in Profile SD-AIX . . . 116
90. SD-AIX Profile Manager . . . 117
91. SD-AIX Profile Manager . . . 118
92. Tivoli/Courier Elements . . . 119
93. Profile Manager (SD-AIX) . . . 121
94. File Package Properties (INed - before) . . . 122
95. File Package Properties (INed - after) . . . 123
96. File Package UNIX Options (INed Before) . . . 124
97. File Package UNIX Options (INed After) . . . 125
98. File Package UNIX Options (INed Remove) . . . 126
99. File Package UNIX Options (INed Commit) . . . 127
100. Profile Manager (SD-AIX) . . . 128
101. File Package Properties (INed) . . . 129
102. Export File Package Definition . . . 130
103. Import File Package Definition . . . 131
104. The INed_After.sh script . . . 132
105. The INed_Remove.sh script . . . 132
106. The INed_Commit.sh Script . . . 132

107. Distribution Path for the INed File Package 133
108. Profile Manager (SD-AIX) 134
109. File Package Properties (INed) 135
110. Distribute File Package (INed) 136
111. Distribute File Package (INed with Subscriber) 137
112. Desktop with Operation Status Messages 138
113. Read Notices 139
114. Notice Group Messages (Courier) 139
115. Profile Manager (SD-AIX) 140
116. File Package Properties (INed) 141
117. Remove File Package (INed) 141
118. Remove File Package (INed from Subscriber) 142
119. Desktop with Operation Status Messages 143
120. Profile Manager (SD-AIX) 144
121. File Package Properties (INed) 145
122. Distribute File Package (INed) 146
123. Commit File Package (INed with Subscriber) 147
124. Desktop with Operation Status Messages 148
125. Profile Manager (SD-AIX) 151
126. File Package Properties (MQSeries) 152
127. File Package Properties (MQSeries) 153
128. File Package UNIX Options (MQSeries Before) 154
129. File Package UNIX Options (MQSeries After) 155
130. File Package UNIX Options (MQSeries Remove) 156
131. The MQSeries befmqm.sh Script 157
132. The MQSeries aftmqm.sh Script 158
133. The MQSeries remmqm.sh Script 159
134. Distribution Path for the MQSeries for AIX File Package 160
135. Profile Manager (SD-AIX) 161
136. File Package Properties (MQSeries) 162
137. Distribute File Package (MQSeries) 163
138. Distribute File Package (MQSeries with Subscriber) 164
139. Desktop with Operation Status Messages 165
140. Read Notices 166
141. Notice Group Messages (Courier) 166
142. Profile Manager (SD-AIX) 168
143. File Package Properties (MQSeries) 169
144. Remove File Package (MQSeries) 169
145. Remove File Package (MQSeries from Subscriber) 170
146. Desktop with Operation Status Messages 171
147. Notice Group Messages (Courier) 172
148. Profile Manager (SD-NT) 175
149. File Package Properties (MQSeries NT) 176
150. File Package Properties (MQSeries NT) 177
151. File Package Windows NT Options (MQSeries Before) 178
152. File Package Windows NT Options (MQSeries After) 179
153. File Package Windows NT Options (MQSeries Remove) 180
154. File Package Windows NT (MQSeries Commit) 181
155. The MQSeries NT aftmqm.bat File 182
156. The MQSeries NT commqm.bat File 183
157. The MQSeries NT remmqm.bat File 183
158. Distribution Path for the MQSeries for NT File Package 185
159. Profile Manager (SD-NT) 186
160. File Package Properties (MQSeries NT) 187
161. Distribute File Package (MQSeries NT) 188

162. Distribute File Package (MQSeries NT with Subscriber) . . . 189
163. Desktop with Operation Status Messages . . . 190
164. Notice Group Messages (Courier) . . . 191
165. Profile Manager (SD-NT) . . . 192
166. File Package Properties (MQSeries NT) . . . 193
167. Distribute File Package (MQSeries NT) . . . 194
168. Commit File Package (MQSeries NT with Subscriber) . . . 195
169. Desktop with Operation Status Messages . . . 196
170. Notice Group Messages (Courier) . . . 197
171. Profile Manager (SD-NT) . . . 198
172. File Package Properties (MQSeries NT) . . . 199
173. Remove File Package (MQSeries NT) . . . 199
174. Remove File Package (MQSeries NT from Subscriber) . . . 200
175. Desktop with Operation Status Messages . . . 201
176. Notice Group Messages (Courier) . . . 202
177. Product Install Dialog with Error Messages . . . 203
178. Mount the CD-ROM Drive . . . 208
179. Install Product Window . . . 209
180. File Browser Window. Enter directory name for installation. . . . 210
181. Install Options Window (Postscript Documentation) . . . 211
182. Install Product Window (Inventory Application) . . . 211
183. RDBMS-Specific Data . . . 212
184. Product Install Window . . . 213
185. Install Options Window (PC Scanning Program) . . . 215
186. Install Product Window (PC Scanning Program) . . . 215
187. Product Install Window (PC Scanning Program) . . . 216
188. Install Options Window (Inventory on TME Client) . . . 217
189. Set Resource Roles (Inventory) . . . 218
190. Creating a Profile . . . 219
191. Profile Manager Window (with Profile INVSCAN1) . . . 220
192. Inventory Profile Window . . . 221
193. Customize Inventory Retrieval Window . . . 222
194. Selecting Targets for Inventory Scanning . . . 224
195. Add Scheduled Job . . . 225
196. Read Notices Window . . . 226
197. Notice Group Messages Window . . . 227
198. Notice Message Viewer . . . 228
199. TIVSCAN . . . 229
200. The useradd.mif Script . . . 229
201. Customize Inventory Retrieval (Add Software Signature) . . . 230
202. The useradd.ini Script . . . 230
203. Hardware Inventory (Windows 3.10) . . . 232
204. Select Software Inventory . . . 232
205. Software Inventory (Windows 3.10) . . . 233
206. Hardware Inventory (AIX) . . . 233
207. PC Managed Node Properties . . . 234
208. Managed Node Properties . . . 235
209. Create Query Library . . . 236
210. Create Query . . . 237
211. Inventory Properties . . . 238
212. Fields in INVENTORYDATA View . . . 239
213. Extended INVENTORYDATA View . . . 240
214. The add_ref_ora.sh Script . . . 243
215. The add_ref_system.sh Script . . . 244
216. The get_diff_sw.sh Script . . . 246

217. The list_inst_software.sh Script . . . 247
218. The list_ref_software.sh Script . . . 247
219. Distribute File Package (MQSeries) . . . 250
220. Execute a Query . . . 250
221. Execute a Query . . . 251
222. Distribute File Package (MQSeries with Subscribers) . . . 252
223. Add the Admin Resources to the Policy Region . . . 254
224. Create Profile Manager . . . 255
225. Create Profile Manager ITSO_Users . . . 256
226. ITSO_Users with No Profiles . . . 256
227. Create ITSO_Residents . . . 257
228. ITSO_Users with ITSO_Assignees and ITSO_Residents Profiles . . . 257
229. Editing Default Policies . . . 258
230. Edit Default Policies Window . . . 259
231. Enter a Value for Attribute City . . . 260
232. Edit the Policy Script for Attribute UID . . . 261
233. Populate the Profile . . . 262
234. Get Records from Managed Nodes . . . 262
235. Populate Errors . . . 263
236. ITSO_Residents Populated with Records from rs600024 . . . 264
237. Select Users to Delete . . . 265
238. Delete or Leave Home Directory . . . 266
239. ITSO_Residents Records . . . 266
240. ITSO_Assignees Record After wpopusrs Command . . . 267
241. Create Profile Manager Test User Profile Distribution . . . 268
242. Add Subscribers to Profile Manager . . . 268
243. Set Current Subscribers . . . 269
244. Add Subscribers to ITSO_Users Profile Manager . . . 269
245. Edit Record in User Profile . . . 270
246. Remove all Current Subscribers . . . 271
247. Fill in User Name, Common Login and Common Password . . . 272
248. Default Values are Filled in with Generate Defaults . . . 272
249. Limit the Category List . . . 273
250. Current Subscribers Category . . . 274
251. Change Current Subscribers and Automatic Subscription Behavior . . . 275
252. New User Added to Profile ITSO_Assignees . . . 275
253. Setting Distribution Defaults . . . 276
254. Distributing the Profile . . . 277
255. Open Subscriber Icon . . . 278
256. Subscriber's Copy of ITSO_Assignees . . . 279
257. Resources Managed by Adminstrator User_admininistrator . . . 280
258. ITSO_Assignees@Test User Profile Distribution Profile . . . 281
259. User Record has Been Changed . . . 282
260. Distribute to All Levels of Subscribers . . . 283
261. Open the Icon for Subscriber rs600019 . . . 284
262. Subscriber's Copy of ITSO_Assignees . . . 284
263. Distribution Options from a Managed Node . . . 285
264. When UID Is Changed . . . 286
265. Operation of TME 10 Distributed Monitoring . . . 293
266. Components of TME 10 Distributed Monitoring . . . 294
267. Viewing the Sentry Monitors . . . 295
268. Create a Profile Manager . . . 297
269. Create a Profile Manager . . . 297
270. Policy Region with the New Profile Manager Icon . . . 298
271. Create a Profile . . . 298

272. Create a SentryProfile . . . 299
273. Edit the Properties of a Profile . . . 300
274. Set Message Styles . . . 301
275. Select Monitor Type . . . 302
276. Edit Sentry Monitor . . . 303
277. Set Monitoring Schedule . . . 303
278. Restrictions . . . 304
279. syslog_filter.ksh Script File . . . 306
280. Add Monitor to Tivoli/Sentry Profile . . . 307
281. Set Distribution Actions Window . . . 308
282. filterkill.ksh File . . . 309
283. filtercheck.ksh File . . . 310
284. Defining Managed Resources for a Policy Region . . . 310
285. Set Managed Resources Window . . . 311
286. Create Indicator Collection . . . 311
287. Define the Name of a New Sentry Indicator Collection . . . 311
288. Select Indicator Collection . . . 312
289. Select Web Server Indicator Collection . . . 312
290. Select a Subscriber for the Profiles . . . 313
291. Distribute the Profiles . . . 313
292. Distribute the Profiles . . . 314
293. Components of the Tivoli Enterprise Console . . . 320
294. Installing the T/EC Components . . . 321
295. Configuring Database Options . . . 322
296. T/EC Installation Message . . . 323
297. Configuring an Administrator . . . 324
298. Selecting Administrator Roles . . . 325
299. Creating the Event Console for SNMP_MANAGER . . . 326
300. Creating the Console for SNMP_MANAGER . . . 327
301. Where the Console Will Execute . . . 327
302. The SNMP_MANAGER Desktop . . . 328
303. Root Desktop for T/EC Configuration . . . 329
304. Create a New Rulebase . . . 330
305. Define the NetView Rulebase . . . 330
306. Copying the Rulebase . . . 331
307. Defining the Target of the Copy Operation . . . 331
308. Activate the NetView Rulebase . . . 332
309. Activate the Rule Base . . . 332
310. Open the Event Server Icon . . . 333
311. Import a New Class Definition . . . 334
312. Import the Class Definition File nvserverd.baroc . . . 335
313. Assigning New Event Groups . . . 336
314. Assigning a New Source . . . 337
315. Assigning New Event Groups . . . 337
316. The Event Filter Definitions . . . 338
317. T/EC Event Group Selection . . . 339
318. Assign Event Groups to SNMP_MANAGER Console . . . 339
319. Setting T/EC Server Parameters . . . 340
320. T/EC Console Message Limit Controls . . . 341
321. The T/EC Event Console Group Display . . . 341
322. The T/EC Event Display . . . 342
323. View Message Details . . . 342
324. Group Display for Paul . . . 343
325. Altering the Sort Order for an Event Console . . . 343
326. Altering the Display Fields for an Event Console . . . 344

327. The Customised Events Display . . . 344
328. Navigator Window . . . 345
329. T/EC Task Definitions . . . 346
330. The Message Appearing on the Console . . . 347
331. Edit Criteria Fields . . . 348
332. Final Automated Task Configuration . . . 348
333. The New Task TEC_NetFinity . . . 349
334. NetFinity Web Page . . . 351
335. The TEC_NetFinity Task Definition . . . 352
336. The TEC_NetFinity Task . . . 353
337. Example of wtdumprl Output . . . 354
338. Result of the wtdbspace Command . . . 354
339. Defining the LOGFILE Source . . . 358
340. Define Filters for Event Group 'System Log' . . . 359
341. Assign Event Group 'System Log' to an Event Console . . . 359
342. Su_Success Event Sent from Logfile Adapter . . . 360
343. Editing Policy Region Managed Resource List . . . 364
344. Adding Adapter Configuration Profiles to the Policy Region . . . 364
345. Assigning ACF Administration Roles . . . 365
346. Create a New ACF Profile . . . 365
347. Selecting the Adapter Type for an ACF Profile Entry . . . 366
348. Updating the Logfile Adapter Configuration . . . 366
349. Defining the Destination for Additional Adapter Files . . . 367
350. Available Event Groups . . . 369
351. Events with Source NV6K and Class OV_Node_Down . . . 370
352. Comparing the Two NetView Event Adapters . . . 371
353. Configuring the NetView Adapter Using Smit . . . 371
354. The nvserverd.baroc File . . . 372
355. The TEC.rs Ruleset Definition . . . 374
356. The Node Down Definition from TEC.rs . . . 375
357. Sentry Monitor Components . . . 377
358. Monitor for netmon Status with T/EC Event Forwarding . . . 378
359. Sentry Events in the Console Display . . . 379
360. wtdumprl Output for an NT Logfile Event . . . 380
361. NT Events from the NT Adapter . . . 381
362. Edit Rules . . . 385
363. New Ruleset . . . 385
364. Creating the New Rule . . . 386
365. Entering Event Conditions . . . 386
366. Action Definition . . . 387
367. Rule Definition . . . 387
368. Compiling the Rulebase . . . 388
369. Load the Ruleset . . . 388
370. The Working Ruleset . . . 389
371. Node Up/Down Rule Definition . . . 390
372. Activate Tracing . . . 392
373. Rule Trace for nvupdown.rls . . . 392
374. Messages of Interest in Logfile AMQERR01.LOG . . . 394
375. BAROC Class Definitions for MQ Failure Messages . . . 395
376. Mapping Logfile Messages to Baroc Slots . . . 396
377. Additional Format Description for the Logfile Adapter . . . 397
378. Debug Information from the Logfile Trace File . . . 399
379. Create a New Event Group MQS Log . . . 399
380. Define a Filter for the Event Group MQS Log . . . 400
381. Assign the Event Group MQS Log to Your Event Console . . . 400

382. Start and Stop the Static Channel . . . 401
383. Events Generated from T/EC Logfile Adapter . . . 401
384. Event MQS_Channel_Channel_started . . . 402
385. TEC_Start_MQS_Channel.ksh Shell Script . . . 403
386. Extended Class Definitions for MQ Failure Messages . . . 407
387. Default Policy Region for T/EC . . . 407
388. Default Task Library T/EC Tasks . . . 408
389. MQS Task Start_MQS_Channel . . . 408
390. Executable for the AIX 3.2.5 Platform . . . 409
391. T/EC Rule to Start MQ Channel Recovery Automatically . . . 409
392. T/EC Server is not Available for Event Consoles . . . 410
393. Unprocessed Event in the T/EC Reception Log . . . 410
394. Events Generated through Executing Task from Rule Engine . . . 411
395. Event MQS_Channel_Trigger_changed . . . 412
396. Event MQS_Channel_Ping_failed . . . 413
397. Systems Monitor Event Description . . . 415
398. MLM Trap in the NetView Event Display . . . 416
399. Using the MIB Browser to Discover Object IDs . . . 417
400. Extract from the tecad_nv6k.oid File . . . 417
401. Baroc Definition for Additional MLM Trap . . . 418
402. MLM Additions to the CDS File . . . 418
403. Output From wtdumprl . . . 419
404. SM6K_ARM Events . . . 419
405. SM6K_ARM Message . . . 420
406. TEC.rs for All 8260 Alerts and NetView Node Status Events . . . 421
407. TEC.rs Trap Settings Definition for all 8260 Traps . . . 422
408. 8260 Event in the T/EC Display . . . 423
409. The Tivoli/Plus Icon . . . 427
410. Remedy Plus Module Icon . . . 429
411. Remedy Plus T/EC Installation . . . 429
412. Remedy Plus T/EC Installation . . . 430
413. T/EC Install Messages . . . 430
414. T/EC Group Configuration . . . 430
415. Add Remedy Server . . . 431
416. Subscriber Information for ARS Servers . . . 431
417. ARS Collection . . . 432
418. Customized TroubleTicket.sh Script . . . 432
419. Creating a New Trouble Ticket . . . 433
420. The Detailed Description of the Event . . . 433
421. List of ARS Filter Actions . . . 434
422. Editing an ARS Filter Action . . . 434
423. The ADSM Collection . . . 436
424. Defining Package Details for OS/2 . . . 437
425. Defining Package Details for an RS6000 Client . . . 437
426. Distribute Client Software Icon . . . 438
427. Generated File Packages . . . 438
428. File Package Definition with Log File Option Enabled . . . 439
429. T/EC Event Console Group Definition . . . 440
430. T/EC Events Created from the ADSM Module . . . 440
431. Net.Commander Collection Icon . . . 442
432. Net.Commander Policy Region Contents . . . 442
433. Create a New HTTP Server . . . 443
434. Create an HTTP Server . . . 444
435. Adding a Subscriber to the WWW Servers Profile Manager . . . 445
436. Subscribers for Profile Manager: WWW Servers . . . 445

437. Tasks and Jobs for WWW Servers . . . 446
438. Sentry Monitoring Schedule . . . 447
439. Error Response Settings in Netscape . . . 448
440. Baroc Definition for Audit Subsystem . . . 452
441. Logfile Format File Extension for the Audit Subsystem . . . 453
442. Listing the Contents of the Exported Task Library . . . 456
443. Exported Task Library Definition . . . 456
444. Task Library Definition for NetView_Tasks . . . 458
445. The New Task Library Appears . . . 460
446. The New Task Library Has a Task in It . . . 461
447. Task Execution Dialog . . . 462
448. Dialog Prompts for Command Input . . . 463
449. Choice List for Hostname . . . 463
450. Check Managed Resources . . . 464
451. Create a Profile Manager . . . 465
452. Define Execution Targets for Job . . . 466
453. Job and Task Icons in a Task Library . . . 467
454. Selecting Managed Resource Policies . . . 468
455. The tl_def_man_nodes Method Script . . . 469
456. The tl_def_prof_mgrs Method Script . . . 470
457. Create a New Policy Region . . . 472
458. Create a New Policy Region . . . 472
459. New Policy Region Icon . . . 473
460. Task Library Moved to New Policy Region . . . 473
461. Invoke the New Policy Object . . . 474
462. Reduced Target List Generated by New Policy Object . . . 475
463. Selecting Validation Policy . . . 478
464. Validation Error Message . . . 478
465. Column Headings Show the Names of the User Profile Properties . . . 479
466. The List of Attributes Equate to the User Profile Properties . . . 480
467. AIX4_Authentication Column Added with Value of Compat . . . 481
468. List of Subcategories for All Categories . . . 482
469. Select UNIX Subcategories . . . 482
470. Properties in the UNIX Login Subcategory . . . 483
471. Examples of Different Layouts for Property Fields . . . 484
472. Preview New Panel . . . 486
473. Panel Defined with Extra Fields . . . 488
474. AIX4 Category Now Available . . . 489
475. AIX4 Login Authentication Panel . . . 489
476. UNIX Password Panel Including the Login Authentication Field . . . 490
477. Action Script login_authentication.sh . . . 492
478. Registration File for nvevents . . . 496
479. Ruleset Editor . . . 498
480. Node Down Trap Setting . . . 499
481. Completed Example Ruleset . . . 500
482. NetView for AIX Configuration Panel . . . 501
483. Configuration for Tivoli Panel . . . 502
484. General_tasks Registration File . . . 503
485. NetView_tasks Registration File . . . 504
486. general_tasks.sh Shell Script . . . 506
487. Run_general_tasks Shell Script . . . 507
488. NetView_tasks.sh Shell Script . . . 508
489. Run_netview_tasks.sh Shell Script . . . 509
490. Code Excerpt to Prompt for Input in an Xterm . . . 510
491. The TME_fields File . . . 511

492. NetView for AIX Collections, Profile Managers 512
493. NetView for AIX Collections, Policy Regions 512
494. NetView for AIX Collections Definition 513
495. Resolved List of Nodes in Policy Region Collection 514
496. Collection Submap for All TME Nodes 514
497. Shell Script create_tme_collections.sh 515
498. Program set_tme_field.c 517

Tables

1. Authorization Roles for Administrators 8
2. Administrator Function to Policy Region 15
3. Roles on Resource Level 71
4. Roles on TMR Level 71
5. Performance Data Collection Phases 79
6. Data Analysis 80
7. Overall Memory Requirements 80
8. Setting File Package Profiles 108
9. Defining and Deleting File Packages 108
10. Performing File Package Operations 108
11. Distribution Actions on Profile 308
12. Sentry Monitor Planning Table 315
13. T/EC Troubleshooting 360
14. Inheritance of Slot Definitions 395
15. Components Required by Net.Commander on Different Servers 441
16. Classes 449

© Copyright IBM Corp. 1996

Preface

This redbook provides detailed information about the Tivoli Management Environment (TME) on AIX. Information is included about how to install and use TME 10 systems management products with the AIX operating system.

The book uses practical examples to illustrate the basic installation and configuration tasks for all of the currently available products that use the Tivoli framework. It also shows some examples of how the base function can be extended, by means of Tivoli/Plus modules and customization. In addition there are more detailed customization examples that exploit some of the programming interfaces that TME provides.

This redbook was written for system administrators and support staff who need to know more about the Tivoli Management Environment for systems and network management. When combined with the Tivoli product documentation it provides a good source for information about planning, implementing and maintaining a TME configuration. The redbook is also of benefit to consultants and managers who need to gain broad understanding of the capabilities of the Tivoli Management Environment for planning and design purposes.

System support professionals and administrators will find the step-by-step examples within the book are an invaluable guide when starting out on a new TME-related project. Further redbooks will be produced in the ensuing months which will expand on these examples in specific areas of detail.

How This Redbook Is Organized

This redbook contains 555 pages. It is organized as follows:

- Part 1, “Introduction, Installation and Base Platform Capabilities”

 This section introduces the concepts on which the Tivoli Management Environment (TME) is based, and describes how it is installed and configured.

 - Chapter 1, “Introduction to the Tivoli Management Environment”

 This chapter explains the concepts behind the Tivoli Management Environment as well as the Tivoli management principles.

 - Chapter 2, “Planning and Installing the Tivoli Management Platform”

 This chapter describes the configuration tasks and provides additional information on various tasks and topics.

 - Chapter 3, “Configuring the TME Management Platform”

 This chapter guides you through the steps necessary to connect TMRs, to set up policy regions, and to create TME administrators.

 - Chapter 4, “Task Libraries, Tasks and Jobs”

 This chapter describes the TME built-in services and the creation of task libraries, tasks, and jobs.

- Part 2, “Deployment Applications”

© Copyright IBM Corp. 1996

This section describes the TME core applications that are used for deploying distributed applications, primarily for software distribution and user administration.

 - Chapter 5, “Tivoli/Courier”

 This chapter discusses the features and uses of the Tivoli/Courier product for deployment management (software distribution).

 - Chapter 6, “Tivoli/Inventory”

 This chapter discusses the features and uses of the Tivoli/Inventory product for deployment management (software distribution).

 - Chapter 7, “Tivoli/Courier and Tivoli/Inventory Interoperability”

 This chapter describes how to query Tivoli/Inventory data to select end points for Tivoli/Courier software distribution.

 - Chapter 8, “TME 10 User Administration”

 This chapter discusses the features and uses of the Tivoli/Administration product for central management of remote systems for the configuration of users and groups, host configurations (including NIS), mail aliases, and a number of standard system configuration files. This chapter concentrates on user management.

- Part 3, “System Monitoring and Event Handling Applications”

 This section describes the TME core applications that are used to monitor distributed systems and provide centralized event handling.

 - Chapter 9, “TME 10 Distributed Monitoring”

 This chapter describes the Tivoli/Sentry product, which is used for monitoring the performance and behavior of remote systems.

 - Chapter 10, “Introduction to the TME 10 Enterprise Console,” Chapter 11, “Tivoli/Enterprise Console Adapters,” and Chapter 12, “More Advanced TME 10 Enterprise Console Customization”

 These chapters introduce the TME 10 Enterprise console. They also explore the different sources of T/EC events and the facilities for extending the base capabilities of T/EC.

- Part 4, “Integrating Management Applications into TME”

 This section describes how the core applications within TME are integrated together to provide solutions for systems management problems. It also discusses some of the facilities for extending TME capabilities.

 - Chapter 13, “Tivoli/Plus Modules” and Chapter 14, “TME 10 Net.Commander”

 These chapters describe pre-packaged solutions that use the TME components to deliver systems management functions

 - Chapter 15, “Adding Function to the TME Desktop”

 This chapter gives detailed examples of some of the facilities for extending TME platform capabilities, namely Task Library Language (TLL), Policy Methods and Application Extension Facility (AEF).

 - Chapter 16, “Integrating NetView for AIX with Other TME Functions”

 This chapter describes how NetView for AIX can be further integrated with the other TME 10 components.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the Systems Management and Networking ITSO Center, Raleigh.

Bernhard Bitzel	IBM Germany
Paul Fearn	IBM UK
Alessio Frenquelli	IBM Australia
Lynn Holmes	IBM UK
Bernd Kammholz	IBM Germany
Guenther Mayerhoffer	IBM Germany
Wolfgang Milchram	IBM Austria
Michiko Mohri	IBM Japan
Sanath Perera	IBM Australia
Guenter Rieker	IBM Switzerland

The advisors of this project were:

Wolfgang Geiger	ITSO-Raleigh Center
Rob Macgregor	ITSO-Raleigh Center
Fred Plassman	ITSO-Raleigh Center
Dave Shogren	ITSO-Raleigh Center

Thanks to the following people for their valuable contributions to this project:

Paul Braun, Shawn Walsh, David Boone, and numerous additional staff from the Systems Management and Networking ITSO Center, Raleigh.

Karl Gottschalk, Dave Hart, Walt Giroir, Richard Buckman, Greg Kattawar
Tivoli Systems

Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us a note at the following address:

redbook@vnet.ibm.com

Your comments are important to us!

Part 1. Introduction, Installation and Base Platform Capabilities

© Copyright IBM Corp. 1996

Chapter 1. Introduction to the Tivoli Management Environment

The architecture of the Tivoli Management Environment is organized around the principle that certain systems management processes must be standardized across the enterprise to ensure a consistent, policy ruled management of all systems within that enterprise. But contrary to other designs that only focus on defining the architecture, the Tivoli Management Environment also provides a set of fundamental tools for managing client/server systems. Figure 1 shows the TME architecture model.

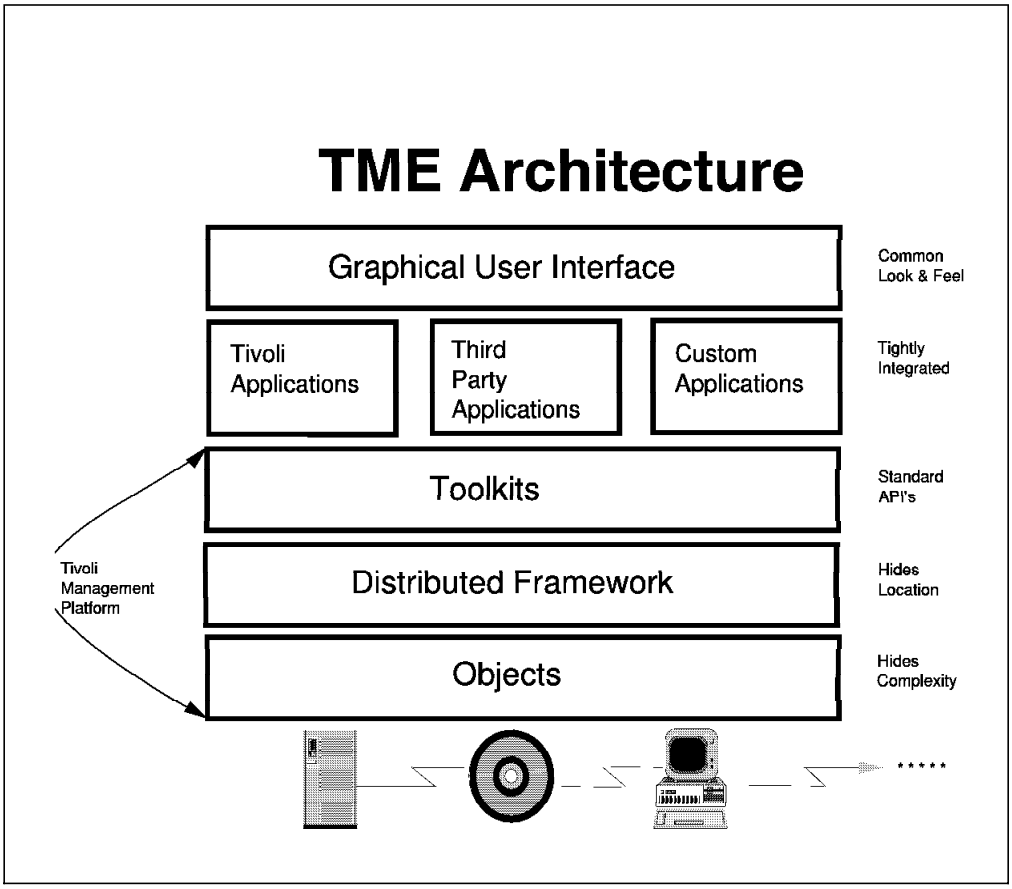


Figure 1. TME Architecture

In order to understand the later chapters in this book, which explain our specific setup of the Tivoli Management Environment at the ITSO, as well as the usage and the capabilities of all the TME products, it is essential to understand the concepts and expressions used when working in a Tivoli managed environment.

This chapter explains the concepts behind the Tivoli Management Environment as well as the Tivoli management principles.

© Copyright IBM Corp. 1996

1.1 Components of the Tivoli Management Platform

The Tivoli Management Platform (TMP) provides the foundation for managing resources in a distributed environment in two ways:

1. As the basic administrator′s view into the network, as well as with a set of tools that are applicable across functions and applications
2. As the run-time platform for TME applications

The platform provides the ability to create administrators and assign them authorization roles based on the systems management tasks they are to perform. It also includes a notification facility that tracks changes made in the TME and reports these changes to the appropriate administrator.

The Tivoli Management Platform provides a set of systems management services that enable you to install both the platform and selected applications on multiple, heterogeneous systems. Once installed and configured, the platform provides a robust foundation for managing TME resources, policies and policy regions.

The following list provides you with an overview of the functionalities available after the initial installation of the Tivoli Management Environment:

- An administrator′s desktop which gives access to all or part of the management functions. This includes the functions provided by all of the add-on modules such as Tivoli/Admin, Tivoli/Courier, Tivoli/Inventory and Tivoli/Plus modules.
- Bi-directional communication link between multiple policy regions. This allows distributed management from one central source.
- Graphical installation facility to easily install the Tivoli components and modules on the TME servers and clients.
- An ability to catalog managed nodes and install the TME client code on remote clients from the TME server.
- Ability to define multiple administrators, define their privileges and customize their desktops.
- Ability to define tasks. That is assigning an icon to an executable and specifying the conditions for its execution (for example, a UNIX shell script), then executing the program by double-clicking on the icon.
- Some basic functions on UNIX managed nodes such as view properties and run X-term.
- Ability to define policies for the TME framework.
- The capability to visually organize resources into logical groups. For example, a policy region ″Accounting″ can hold all managed systems used by the Accounting department.
- The scheduler facility for controlling scheduled jobs initiated by the Tivoli applications.
- A graphical user interface to perform certain TME administration tasks. For example, to perform a TME database backup. (The full set of functions for administering the TME environment is also provided by a set of commands.)
- A notice icon for all Tivoli messages. These messages can be filtered and assigned to particular administrators.

1.1.1 TME Resources

TME resources correspond to the systems, devices, services and facilities in a distributed system. Some examples of TME resources are workstations, software products, files and directories and people such as users or administrators.

After a resource management application is installed (for example, Tivoli/Admin, which is used to manage users and groups), the Tivoli Management Platform provides you with a single logical view of all resources managed by that application. It also enables you to make changes to these logical resources and updates the actual system resources when the corresponding TME resources get updated.

1.1.2 Tivoli Management Region (TMR)

To meet the demands of managing potentially thousands of geographically dispersed resources, the Tivoli Management Environment provides the capability to logically partition an installation into multiple connected Tivoli Management Regions (TMRs). Each TMR has its own server for managing local clients.

Tivoli Management Regions can be connected together to coordinate activities across the network. This enables large scale systems management activities and provides the ability for *remote site management* and *remote site operation*.

TMR inter-region connections are *directed*, meaning that the connections can either be *one-way* or *two-way*. In a two-way connection each of the TMRs is aware of the existence of the other TMR. Information exchanges about system resources can occur in both directions. In a one-way connection, only one TMR has knowledge of the other, so information is exchanged from the managed system only.

One-way connections are useful where a central site is responsible for administering remote sites, and none of the remote sites have any need to manage resources at the central site or at other remote sites. Each remote site could also have its local operator who might be responsible for managing day-to-day operations on local resources, while the connection from the central site is used for more global updates across the company, such as new versions of an application.

Two-way connections are useful in a variety of situations, including that of a very large local area network that is logically partitioned. This allows you to spread the management server load across multiple TME servers. In addition, two-way connections are needed when two or more TMRs have systems management staff who need to have access to resources at other TMRs.

1.1.3 Policy Regions

A *policy region* is a collection of TME resources that are controlled by a common set of policies. In practical terms this means that policy regions define the boundaries of the authority of each system administrator.

Policy regions provide a mechanism for organizing and managing system resources in a hierarchical structure. Administrators can be assigned different roles for each policy region therefore providing a convenient way to provide restrictive access to the management functions and data. Policy regions also provide a logical view of the organization. Policy regions are often created to

represent a management domain or the geographical organization of your company.

For example, you can set up policy regions to reflect your company's departmental structure. Let's assume your company has two departments: Administration and Sales. You could create two policy regions (Administration and Sales), each containing the resources (computers, databases, applications) that belong to that department. As you define TME administrators you give them *authorization roles* within a policy region to control what level of control they have over the resources within the region.

You could then further subdivide the resources in each of these policy regions by creating *subregions*.

1.1.4 Policies

Policies are rules that control the management of TME resources, such as saying that all users must have passwords. Traditionally these rules take the form of written procedures, guidelines or shell scripts. For many environments policies are simply a set of conventions that may or may not always be followed.

With Tivoli Management Platform you have a policy facility that enables you to encode acceptable policies and values for individual resources. As a result, the rules for managing TME resources can now be enforced. This allows the delegation of systems management tasks, knowing that only changes within the constraints of the defined rules and policies can be made.

1.1.5 Profile Endpoints

A *profile endpoint* is a system resource such as a managed node or an NIS domain that is the final destination of a profile.

1.1.6 Managed Nodes and PC Managed Nodes

When the Tivoli Management Environment is installed, a *managed node* resource is created for each client added to the TMR. This is also true for *PC managed node* resources that are not already clients of a NetWare managed site.

Managed nodes and PC managed nodes are basically the same; they are both managed resources in the TME database that represent a single machine. The main difference is that a PC managed node has a TME agent installed, whereas a managed node has the Tivoli Management Platform installed.

Managed nodes represent machines running a supported version of UNIX or Windows NT. PC managed nodes are TME resources representing machines running Windows, Windows 95, Windows NT, DOS, OS/2, and NetWare.

1.1.7 Profiles

A *profile* is a collection of application-specific information. The information in a profile is specific to the particular profile type.

For example, a user profile will contain information such as the user names, login names, etc., whereas a software distribution profile contains names of software files to be installed, names of pre-installation scripts, etc. Managed resources (which can be systems, but also databases, applications or even other

profiles) subscribe to these profiles. Therefore, all systems subscribing to a particular user profile would have identical user definitions.

For a practical example, let us assume that a new user starts working in the Accounts department, and needs a user ID created on multiple UNIX systems used by this department.

The steps to perform this in TME 3.0 would be as follows:

1. Open a Profile Manager that manages users.

 Profile managers serve as containers that link profiles with their subscribers.

2. Open the Profile containing the user IDs for the Accounts department.

 A profile contains all of the information for a specific managed resource (in this case, users). It is possible to have multiple user profiles, each for a different department.

3. Add the new user definition to the profile and save.

 This updates only the profile in the TME database; the actual UNIX machines in the Accounts department will not show any changes.

4. Distribute the profile to all of its subscribers.

 Our subscribers would, in this case, be all of the UNIX systems in the Accounts department.

1.1.8 Profile Managers

A *profile manager* is a container for multiple profiles, which are subscribed to as one unit by individual profile endpoints.

A profile manager contains a list of subscribers to which the profile data can be distributed. The subscribers can be managed nodes or PC managed nodes. In addition to this, other profile managers can subscribe to other profile managers.

Profile managers control the handling of any differences between the local profile records and the original profile records.

When a profile manager distributes a profile copy to a subscriber, the source data is merged with the subscribers local database, unless otherwise specified. The database that results in the merging of the local profile data and the source profile data is then passed on down the hierarchy. This process continues recursively until a profile reaches an endpoint. When the endpoint is reached, the data is passed to the client's object dispatch broker that then actually modifies the system files.

1.2 Tivoli Administrators

A *Tivoli administrator* is a system administrator who has been established as a Tivoli Management Environment administrator. The installation of the Tivoli Management Environment is performed by the root user. Therefore, root becomes the initial Tivoli administrator. After TME is installed, other non-root administrators can be defined and given roles in the TME.

Tivoli administrators can perform systems management tasks and manage various regions. Systems management tasks can be delegated to different system administrators by:

- Assigning individual authorization roles
- Moving or copying policy regions between desktops
- Moving or copying system resources between policy regions

With *authorization roles*, you can assign different roles to administrators for various policy regions. This allows them to do certain systems management tasks within the policy regions for which they are responsible, yet still limits their access to other system resources.

This ability to delegate authority gives senior administrative personnel complete control over who can perform specified operations on different sets of resources.

When an administrator starts a Tivoli desktop, it displays those Tivoli policy regions and resources that the administrator has authorization to manage. In order to assign roles and to delegate management for certain resources, the respective policy region icon can be dragged and dropped onto the administrator's desktop.

1.2.1 Authorization Roles

When new administrators are added, they are assigned certain *administration roles*. These administration roles can be altered at any time by an administrator with the *super* or *senior* role.

An administrator can be assigned authorization roles on TMR or resource level:

- Authorization roles given in a TMR enables an administrator to perform actions that may affect resources anywhere in the local TMR. These roles (except super) map across the boundaries of all two-way connected TMRs.
- Authorization roles on the resource level enable an administrator to perform management tasks over that specific resource. If, for example, an administrator is responsible for managing the systems in the Accounts department, the administrator would most likely have the senior role in the Accounts region, and have the user role in the Sales region. This is because managing the systems in the Sales department is not part of an administrator's responsibilities.

The possible authorization roles for administrators defined in the Tivoli Management Platform are shown in Table 1.

Table 1 (Page 1 of 2). Authorization Roles for Administrators

Role	Authorization
super	Required only for high security operations such as: • Connect/disconnect TMRs • Change license key
senior	Required for configuration and policy tasks such as: • Creating administrators • Setting policies

Table 1 (Page 2 of 2). Authorization Roles for Administrators	
Role	**Authorization**
admin	Required for general systems administration tasks such as: • Pushing a file package • Adding a user item to a profile
user	Allows limited operations that do not affect configuration information: • Required to bring up desktop
restore	Required to restore TME databases: • The restore role is required in the TMR that contains the TME server and the clients to be restored.
backup	Required to back up TME databases: • The backup role is required in the TMR that contains the TME server and the clients to be backed up.
install-product	Required to install new applications into the local TMR
install-client	Required to install new managed nodes within policy regions

Authorization roles are not hierarchical

The authorization roles provided by the Tivoli Management Platform are mutually exclusive, not hierarchical. This means that each role provides its own set of privileges and no role provides the privileges of any other role.

1.2.2 Notice Groups

A notice is generated when a TME management operation is performed, and the notices are sent to an application or operation-specific *notice group*. A notice group stores and distributes messages pertaining to specific TME functions. For example, the TME Administration notice group receives notices from such operations as creating an administrator or changing the set of resources managed by a policy region.

The Tivoli Management Environment provides a default set of notice groups:

- TME Scheduler

 Contains notices related to the operation of the scheduler

- TME Administration

 Contains notices related to the general functions of the Tivoli Management Environment such as installation of applications, management of TME administrators, etc.

- TME Authorization

 Contains notices related authorization errors, or to changes in administrator roles, etc.

- TME Diagnostics

Contains notices generated by maintenance operations such as running the wchkdb command that verifies and repairs the TME database.

Notice Groups are TMR specific

In a Tivoli Management Environment that consists of multiple connected Tivoli Management Regions, each TMR controls its own set of notice groups. This means that if you want to receive notices from a remote TMR, you have to explicitly subscribe to the remote TMR's notice groups.

1.3 A Practical Example of a TME Management Concept

The main structures in your Tivoli Management Environment are as follows:

- Tivoli Management Regions (TMR)
- Policy regions
- Profile managers

Tivoli Management Region:

To size a TMR, you have to consider various factors, such as the network topology and bandwidth, number of managed nodes, geography, and organizational considerations, such as different departments.

How Many Clients per TMR Server

A single server can support up to 200 managed nodes. PC managed nodes do not count toward this number, and there can be an arbitrary number of clients running on a PC platform, although other load related figures still need to be considered.

Refer to the *Tivoli Management Platform Planning and Installation Guide* for more information on TMR sizing aspects.

For our example let's assume that this customer wants to centrally manage all decentralized systems, except the machines used for software development, which are going to be managed by this department's own administrator. After careful planning the customer decided to implement the TMR structure as shown in Figure 2 on page 11.

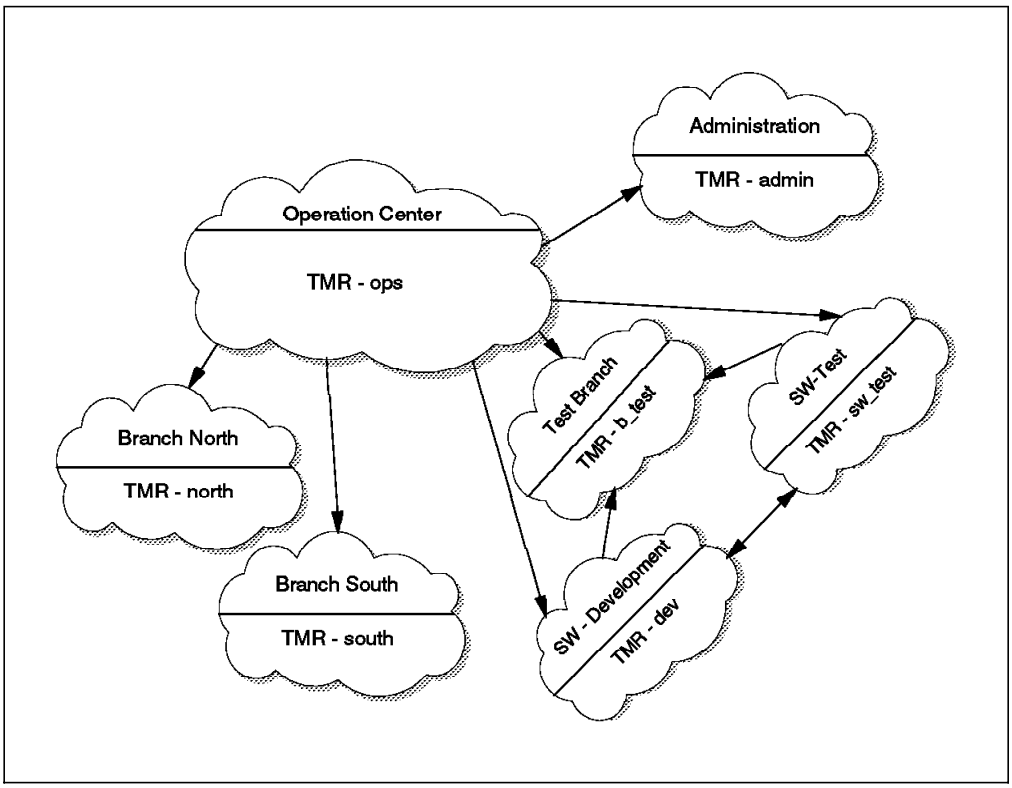


Figure 2. Example of a TMR Structure

- The production environment

 TMR *ops* contains all systems of the Operation Center, TMR *admin* is used for all back-office systems and TMRs *south* and *north* are used for the front office systems.

 The Operation Center's TMR server is *one-way* interconnected with the front- and back-office TMRs and functions as the *managing server*. This means that all resources from the managed TMRs are visible and accessible to the Operation Center, but the Operation Center's resources cannot be accessed from the managed TMRs.

- The development environment

 The Development department has its own three TMRs: TMR *dev* used for software development, TMR *sw_test* for the initial testing, and TMR *b_test* for final testing in a production-like environment. The TMR b_test contains two test environments, representing branches from the southern and the northern hemisphere.

 The TMRs *dev* and *sw_test* are *two-way* interconnected. This means that they can transparently access each other's resources. Additionally they have a *one-way* connection to TMR *b_test*, which allows them to access the resources in that TMR.

 Despite the fact that the Development department is managing its own environment autonomously, the Operation Center has a *one-way* connection to each of these three TMRs. They are mainly used to keep the configuration of the branch test systems synchronized with the actual branch systems in terms of fix levels or software releases.

Policy Regions: As you might recall, our customer wants to manage all of the resources centrally, except for the resources of the Development department. Therefore, we decided to create the policy region structure as shown in Figure 3 on page 12, where the Development department is given its own policy region, containing various subregions. The resources of the production environment were grouped in logical entities according to geography, department, or function.

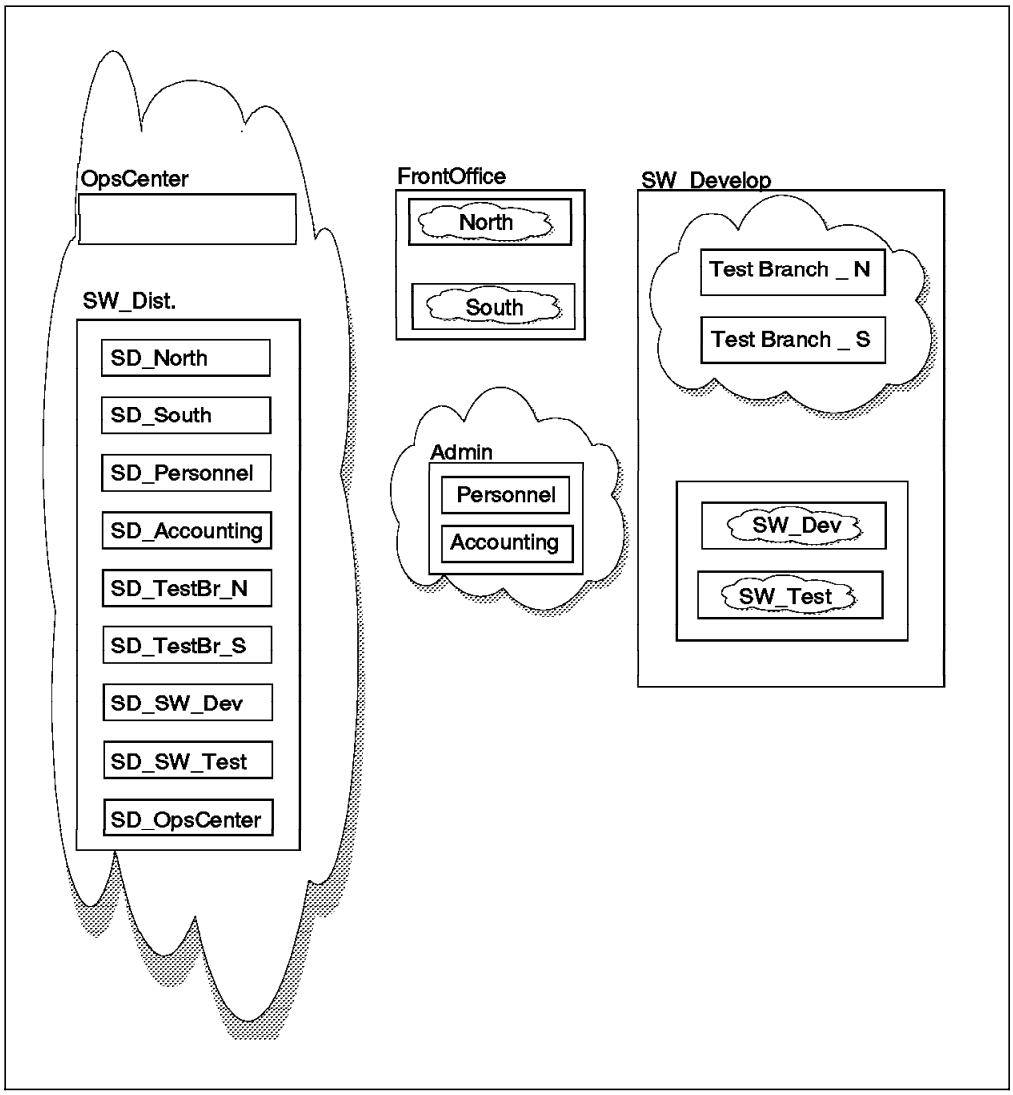


Figure 3. Example of a Policy Region Structure

- FrontOffice

 The policy region *FrontOffice* contains the two subregions *North* and *South*. These subregions contain most of the logical resources, such as profile managers and profiles, and all end nodes (systems) for the respective geographic area.

 In this example, each subregion contains the physical resources of one TMR.

- Admin

 The policy region *Admin* also contains two subregions, namely *Personnel* and *Accounting*. These subregions contain most of the logical resources, such as profile managers and profiles, and all end nodes (systems) for the respective management domain.

In this example, both subregions together contain the physical resources of one TMR.

- SW_Develop

 The policy region *SW_Develop* contains four subregions, namely *TestBranch_N*, *TestBranch_S*, *SW_Dev* and *SW_Test*. These subregions contain most of the logical resources, such as profile managers and profiles, and all end nodes (systems) for the respective management domain.

 In this example, the subregions SW_Dev and SW_Test contain the physical resources of one TMR, whereas the two subregions TestBranch_N and TestBranch_S together contain the physical resources of another TMR.

- OpsCenter

 The policy region *OpsCenter* has no subregion, and contains those logical resources of the Operation Center that are not related to software distribution. Nevertheless, in our example it contains all of the physical resources of one TMR.

- SW_Dist

 The policy region *SW_Dist* contains one subregion for each management domain. These subregions are only used to group the logical resources required for software distribution, such as profile managers, that hold profiles of the type FilePackage and their subscribers.

 In this example, no physical resources are contained in any of these subregions.

Profile Managers: In the example shown in Figure 4 on page 14, we define a profile distribution structure for TCP/IP name resolution.

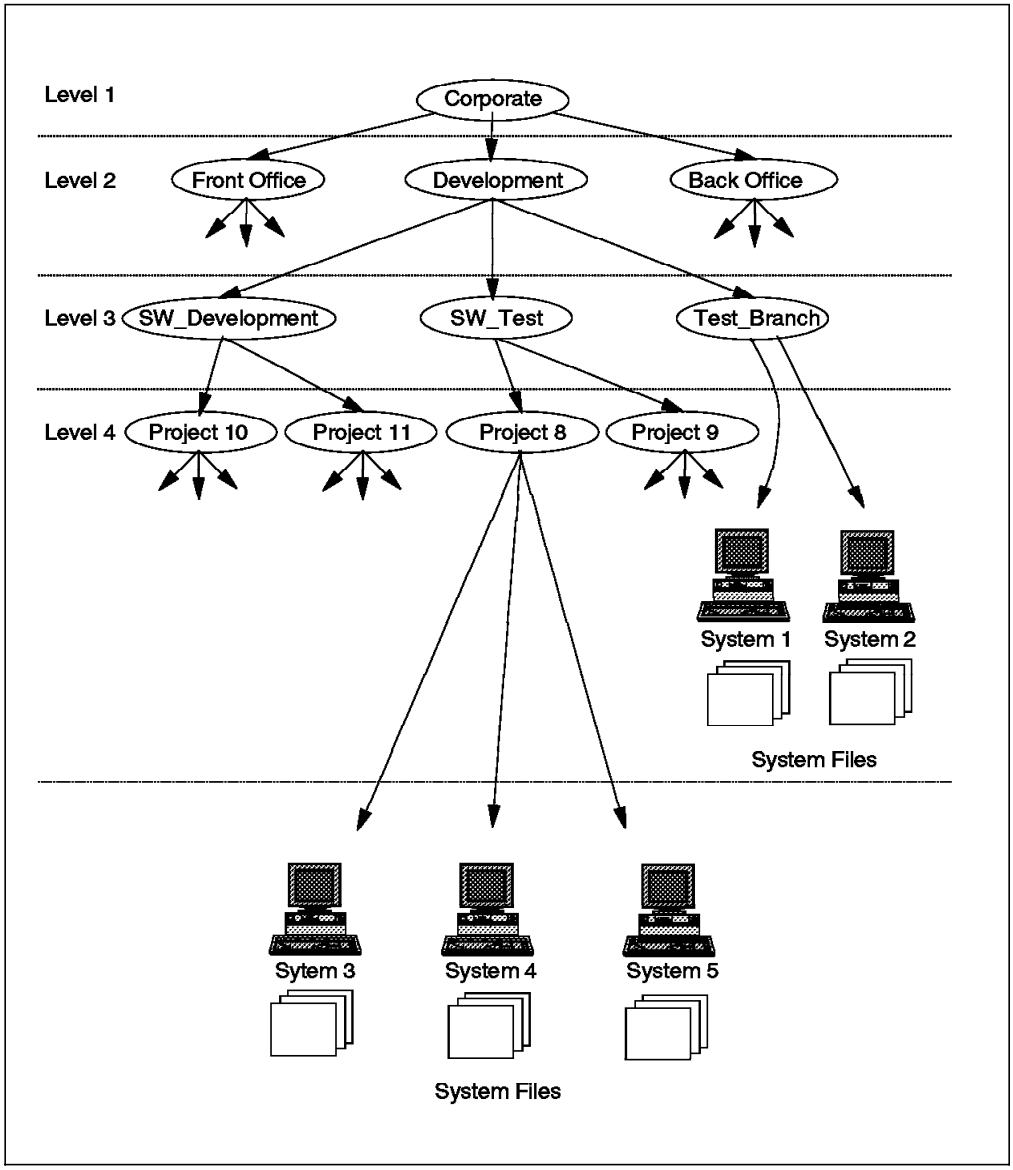


Figure 4. Example of a Profile Manager Structure

Each level in Figure 4 represents a level within the profile distribution hierarchy. This means that profile data from the highest level (in our example Level 1, which represents the Corporate level) is applicable for all subscribers, whereas profile data from an intermediate level is only applicable to subscribers belonging to a specific department or project.

- Level 1

 Profile data defined on this level is applicable to all subscribers of our Tivoli Management Environment.

 On this level we would, for example, define all the gateways that all systems in the whole Tivoli managed environment need to know.

 The only subscribers to the profile manager containing this corporate profile(s) are, in our example, other profile managers.

- Level 2

Profile data defined on this level is only applicable to subscribers that are hierarchically below level 2.

On this level we would, for example, define all gateways or systems that are common to the Development department.

The only subscribers to the profile managers containing these departmental profiles are, in our example, other profile managers.

- Level 3

 Profile data defined on this level is only applicable to subscribers that are hierarchically below level 3. On this level we would, for example, define all gateways or systems that are common to the Software Test group.

 In our example, we find on this level the first profile endpoints subscribing to the profile manager of the Test_Branch. This means that the accumulated host namespace data is going to be applied to the system files on the subscribing endnodes of the Test_Branch, whereas for SW_Test and SW_Develop the merged profile data is being distributed to the profile managers on level 4.

- Level 4

 Profile data defined on this level is only applicable to subscribers that are hierarchically below level 4.

 On this level we would, for example, define all systems that are common to Project 8.

 In our example, there are no further profile managers subscribing to the profile manager on this level. This means that the accumulated host namespace data is going to be applied to all profile endpoints (systems) belonging to the respective project.

Administrators: Administrators can be given various roles on various resources. This permits you to provide a specific administrator with powerful roles in one policy region and with much less powerful role in another region.

In our example, we assigned roles to four different administrator levels to manage our Tivoli Management Environment. The administrator levels in Table 2 do not represent individual TME administrators. Instead, they show the management levels we chose for our example.

Table 2. Administrator Function to Policy Region

Administrator Level	**FrontO**	**Admin**	**OpsC**	**SW_Di**	**SW_De**	**T_BrN**	**T_BrS**
HotShot							
Senior_Prod							
Admin_Prod	√	√	√	√		√	√
Senior_Dev							

Abbreviations used in Table 2:

FrontO Policy Region *FrontOffice*

OpsC Policy Region *OpsCenter*

SW_Di	Policy Region *SW_Dist*
SW_De	Policy Region *SW_Dev*
T_BrN	Policy Region *TestBranch_N*
T_BrS	Policy Region *TestBranch_S*

Authorization Level Symbols sed in Table 2 on page 15:

	Authorization role of *super* or *senior*
√	Authorization role of *admin*

The labels we used to name the administrator levels for our example were chosen freely. They can be anything and can have no special meaning in the Tivoli Management Environment.

- HotShot

 An administrator belonging to this level has a *top level support function* in the whole TME. This administrator is provided with the highest authorization roles for all policy regions.

 The regions TestBranch_N and TestBranch_S are subregions of the top policy region SW_Develop. Therefore, there is no need to individually authorize an administrator for these two subregions, because the authorization roles from a policy region are inherited by that region′s subregions.

- Senior_Prod

 An administrator belonging to this level has got *high-level privileges* in all policy regions making up the production environment. He is provided with no roles for the policy region SW_Develop, because the Development department administers its resources autonomously. The roles for the subregions TestBranch_N and TestBranch_S are necessary to keep the test environment synchronized with the actual production environment.

- Admin_Prod

 An administrator belonging to this level has the roles required to perform the *day-to-day systems management tasks* in all policy regions making up the production environment, such as pushing a file package or adding a user to a profile. The administrator is provided with no roles for the policy region SW_Develop, because the Development department administers its resources autonomously. The roles for the subregions TestBranch_N and TestBranch_S are necessary to keep the test environment synchronized with the actual production environment.

- Senior_Dev

 An administrator belonging to this level has high-level privileges in all policy regions making up the development environment. The administrator is provided with no roles for any policy region belonging to the production environment.

Note: In our example, software distribution is considered a separate, centrally managed task and is performed by the Operation Center for the whole Tivoli managed environment.

Chapter 2. Planning and Installing the Tivoli Management Platform

This book is the result of several projects, whose aims were to familiarize us with the facilities and features of the Tivoli Management Environment and its core applications. Nevertheless, to set up our test environment, we had to go through the same configuration tasks necessary to set up a production environment.

This chapter describes the configuration tasks we performed and provides additional information on certain tasks and topics to the level of detail we found appropriate at the time of writing. By no means does it substitute any of the official Tivoli documentation, which is the most current source of information to the various Tivoli products, and which describes the complete set of functions in full detail.

Note: All of the functions we used to set up and administer our test environment were provided by the *Tivoli Management Platform* and not (as its name might imply) by the *Tivoli/Admin* product, which is used to manage users and groups across multiple UNIX systems.

2.1 TME Management Concept

Setting up and structuring a Tivoli Management Environment is no trivial task and requires a fair amount of planning. Many factors such as the speed and layout of the network, size and geographic location of the administration domains, and type of systems to be managed influence the structure of your Tivoli Management Environment. We therefore recommend that you decide on the sizes and boundaries of your Tivoli Management Regions before you install your TME servers.

Planning Information

Refer to the *Tivoli Management Platform Planning and Installation Guide* for detailed information on the various components comprising the Tivoli Management Environment and for instructions on how to plan your TME installation.

Because there were several projects running concurrently at the ITSO in Raleigh, we structured our Tivoli Management Environment in such a way that all of the Tivoli projects were able to use the same environment without interfering with each other.

This structuring by project (as shown in Figure 5 on page 18) can be compared to the structuring of a productive environment into branches or departments.

© Copyright IBM Corp. 1996

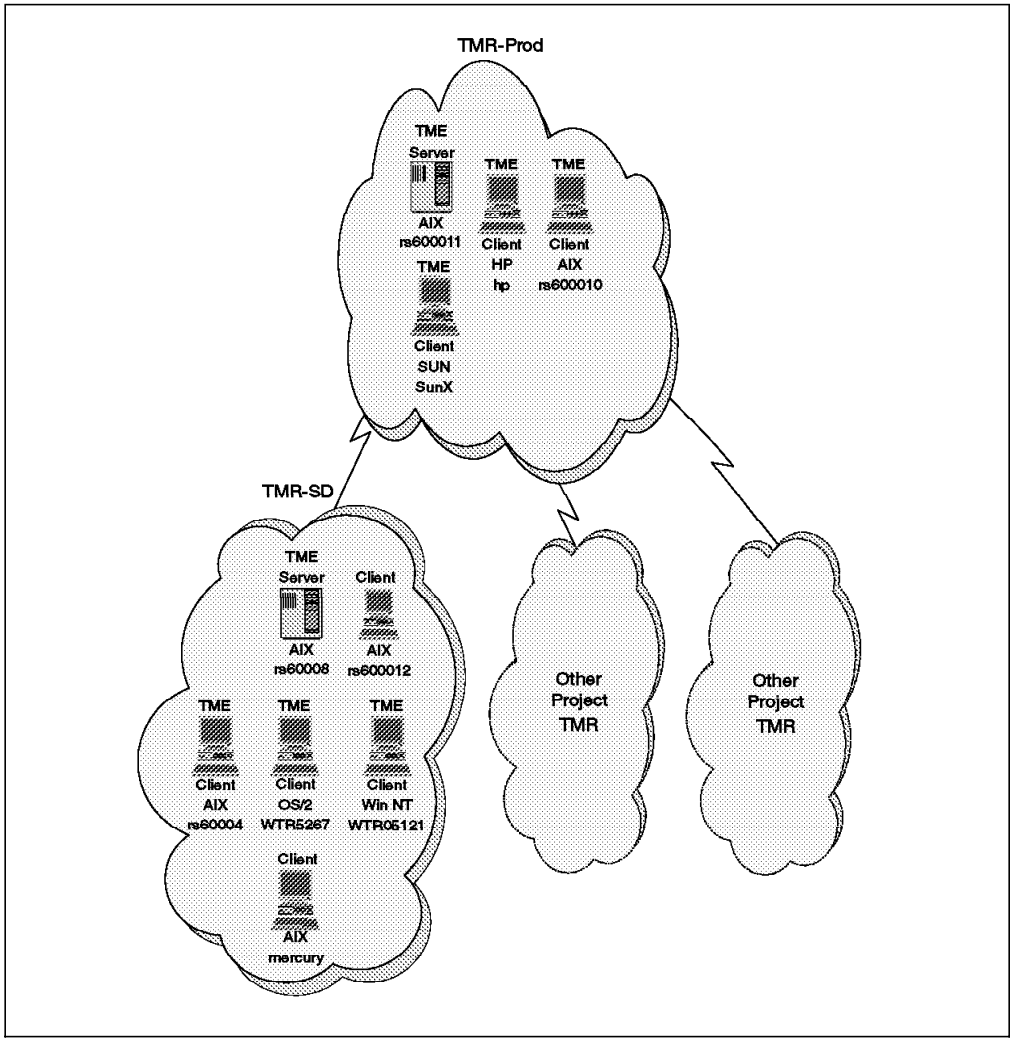


Figure 5. Structure by Project

Tivoli Management Regions are used to partition the Tivoli Management Environment and may be administered locally or remotely from one central site.

Our project required access to all resources in all TMRs; therefore, we used *two-way* connections to interconnect the TMRs. Refer to 1.1.2, “Tivoli Management Region (TMR)” on page 5 for a brief description on the concepts and functions of Tivoli Management Regions.

After setting up the TMRs, you have to create one or more *policy regions*. Figure 6 on page 19 shows the policy regions and *subregions* used for our project.

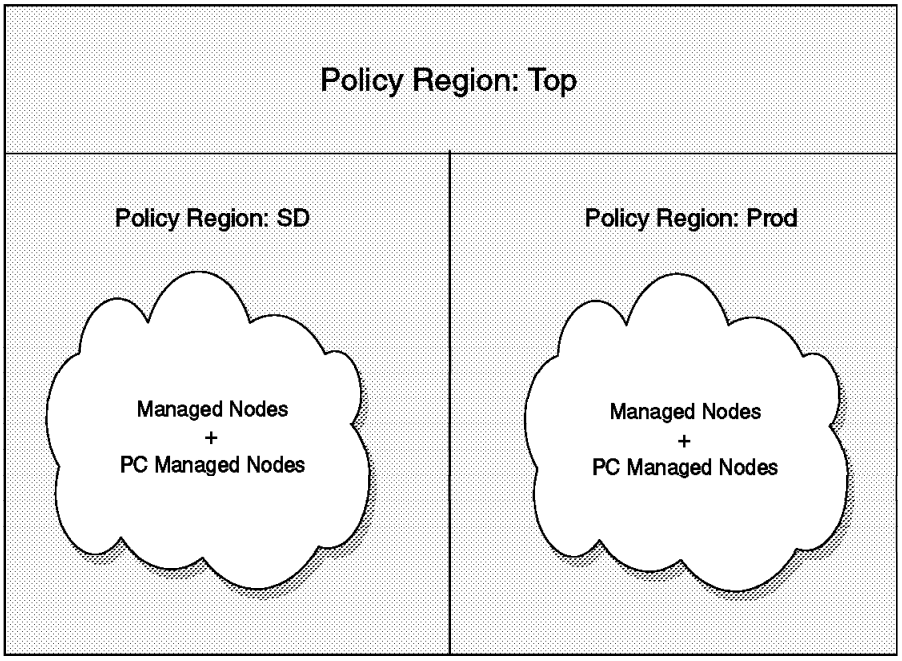


Figure 6. Policy Regions

Policy regions are often created to represent a management domain. One policy region might be sufficient for a simple environment, whereas for a more complex environment multiple regions and subregions may be required.

For our project we created the policy region *Top* containing the subregions *Prod* and *SD*. We then assigned the managed resources to the respective subregions. These resources, for example, were of the type *ManagedNode* or *FilePackage*. Refer to 1.1.3, “Policy Regions” on page 5 for a brief description on the policy region concept.

Note: A policy region is not limited to the boundaries of a TMR. It may include many resources belonging to multiple TMRs or only a few resources within a single TMR.

In a structured TME environment, you will most likely find various administrators with different roles in different management domains. This means that authorization roles are usually assigned depending on the administrator′s responsibilities within a given Tivoli managed environment. If, for example, an administrator′s only responsibility is managing resources of the Accounting department, he or she will be given powerful roles for the Accounting region but read-only privileges for other policy regions. Refer to 1.2, “Tivoli Administrators” on page 7 for a brief description on Tivoli administrators and their administration roles.

Note: For detailed information on the Tivoli administrator subject, refer to the *Tivoli Management Platform User′s Guide*.

2.2 The Environment at the ITSO

Our Tivoli test environment, as shown in Figure 5 on page 18, consists of the following software components:

TME Server OS	AIX 3.2.5
RDBMS	Oracle 7.1.6
Tivoli	TMP 3.0.1
	Courier 3.0
	Inventory 3.0
Managed Nodes	AIX 3.2.5
	AIX 4.1
PC Managed Nodes	OS/2 (only for Courier)
	MS Windows 95
	MS Windows NT 3.5.1

2.3 Security

Network security is an important issue in any client/server environment. The chance of being affected by network intruders increases as the size of the network and the number of connections to other networks increases.

The Tivoli Management Environment provides you with a number of built-in security functions, where you can choose whether and to what extent you want to use the provided functionality.

There are several security issues to consider when setting up a Tivoli Management Environment. This section provides you with the information needed to tailor the Tivoli security functionality to your needs.

2.3.1 Encryption Levels and Passwords

The Tivoli Management Environment provides a facility to encrypt TME security credentials. These security credentials include sensitive data such as *Administrator* or *root* passwords.

The TME provides three levels of data encryption for TME security:

- **None**: The TME does not encrypt any of the TME security data if you choose an encryption level of None. Tivoli recommends that you do not choose this encryption level if your network is not completely secure, all systems on the network are trustworthy and there are no programs or programmers with the ability of network snooping or to tap into the network cabling.

 Generally speaking, the little communication overhead gained by choosing an encryption level of None compared to an encryption level of Simple does not justify the risk of your environment being compromised easily by network intruders.

- **Simple**: The Simple encryption level provides a simple, key-based encryption scheme that protects TME security credentials from casual viewing. The impact on performance of this encryption level is minimal. The communication overhead impact is typically less than five percent. The

Simple encryption scheme is not unbreakable, but a non-trivial amount of time and effort is required to crack the encryption and view any TME security data.

Choosing the Simple encryption level is useful if your network is physically secure and there is no large threat of internal security breaches.

- **DES**: The DES encryption level provides a high level of security, but it also has a significant performance impact. The network overhead is typically more than 12 percent. No encryption scheme is completely unbreakable, but DES security is widely considered to be one of the most secure encryption schemes available.

Simple Encryption is Recommended

Tivoli recommends that you use the Simple encryption scheme for most installations, because this encryption scheme provides a fair level of protection from unauthorized network intrusion at a reasonable cost in terms of communication overhead involved.

Kerberos principals can be used, if required, to authenticate TME administrators without using the DES encryption level.

Inter-Region and Intra-Region Encryption: The Tivoli Management Platform provides you with the ability to set different encryption levels for TME operations within a TMR and between connected TMRs. You can also specify different encryption passwords for intra-region and for inter-region communication, even if both operations use the same encryption level. Additionally, to the different encryption levels, you can specify different TME installation passwords for every TMR, thereby controlling who has the ability to install TME components within this TMR.

You can use any mix of encryption levels and encryption passwords for intra-region, inter-region and TMR installation options.

Note: For more information on the security subject, refer to the *Tivoli Management Platform Planning and Installation Guide*.

2.3.1.1 Setting Encryption Levels and Passwords

When you select an encryption level of Simple or DES you also need to provide an encryption password. The encryption password is used during the encryption process and should be protected like any other password.

This topic provides you with the information necessary to set or change encryption levels, encryption keys and the installation password.

Note: To change the installation password or encryption information you need the super or the senior role.

Installation Password: During the installation of the Tivoli Management Platform you are provided with the Install Tivoli Server dialog as shown in Figure 8 on page 27. Amongst other fields, this dialog contains a field labeled Installation Password. This is the field where you enter the installation password for the local TMR at installation time.

If you want to change the TMR installation password after the initial installation, you do this via the command line.

The following command changes the installation password for the local TMR (you will be prompted to enter both the old and new key):

```
odadmin set_install_pw
```

Intra-TMR and Inter-TMR Encryption: During the installation of the Tivoli Management Platform you are provided with the Install Tivoli Server dialog as shown in Figure 8 on page 27. Amongst other fields, this dialog contains a field labeled Encryption Level. This is the field where you choose the encryption level for your TMR at installation time. The available encryption levels are DES, Simple or None.

If you choose any other encryption level than None, you may also want to provide an encryption key:

- If you *do* provide an Installation Password at installation time, TME uses this password as the installation key (the password to enter for client installations) as the inter-region key (the password required to connect TMRs) and as the intra-region key.
- If you *don't* provide an Installation Password at installation time, TME uses a default key as the inter-region and as the intra-region key. No password is required for client installations.

All encryption levels and encryption keys can be entered or changed via the command line. Here are some examples:

- This command sets the inter-region encryption level:

  ```
  odadmin region set_region_crypt_level simple
  ```

- The following procedure sets the intra-region encryption level. You will have to shut down all client object dispatchers.

  ```
  odadmin shutdown clients
  odadmin set_crypt_level simple
  odadmin start clients
  ```

- The following procedure sets the intra-region encryption key for the client with object dispatcher number 3. You will have to shut down the affected client's object dispatchers. odadmin odlist provides you with the client's object dispatcher number (Disp).

  ```
  odadmin shutdown 3
  odadmin set_ORB_pw 3
  ```

 Change the current directory to the server database directory (for example, /var/local/Tivoli/rs60008.db). Copy the file hostname-od-odb.adj to the file odb.adj in the client's database directory (for example, /var/local/Tivoli/rs60004.db).

  ```
  odadmin start 3
  ```

Installation Password cannot be reset to NULL

You can set or change the installation password at any time. However, if you set an installation password, you cannot set it back to Null.

2.4 Installing the TME Management Platform

The installation of the Tivoli Management Platform can be divided into three tasks:

1. Planning the TME installation
2. Installing TME on a server
3. Installing TME on clients

2.4.1 Planning the Installation

Preparing for a Tivoli Management Platform installation can be divided into tasks related to a TME server installation or to the installation of managed nodes (TME clients).

Managed nodes can again be divided into UNIX or NT managed nodes, which are generally referred to as *managed nodes* or the so called *PC managed nodes*.

The main difference between managed nodes and PC managed nodes is that a PC managed node is a machine that has the TME agent installed, whereas a managed node is a machine with the Tivoli Management Platform installed.

TME servers and managed nodes can be implemented on various platforms. The installation examples provided are for TME servers running UNIX (AIX), managed nodes running UNIX (AIX), and PC managed nodes running Windows 95.

2.4.1.1 Planning a TME Server Installation

License Key: Before you can install the Tivoli Management Environment on a server, you need to obtain a license key. This license key is server-specific and must be entered during the installation process.

The `wnodekey` command generates a unique identifier for your system, which is then used to generate the license key. Refer to the *Tivoli Management Platform Planning and Installation Guide* for current information on how to obtain a license key.

Installation Directories: By default the Tivoli Management Platform will be installed into the following filesystem areas:

- /usr/local/Tivoli

 Binaries, libraries, message catalogs, etc.

- /var/spool/Tivoli

 TME server database

These paths can be changed if required; nevertheless, for our examples we used the default paths.

Required Disk Space: We recommend you install the Tivoli Management Platform into two separate filesystems, one for the binaries and the other for the database.

The initial installation of the Tivoli Management Environment needs approximately 60 MB of disk space in /usr/local/Tivoli and 20 MB in /var/spool/Tivoli.

Name Resolution: Make sure your TME server and all clients (managed nodes and PC managed nodes) are able to resolve each other's hostnames before you start the installation.

Reinstallation of the TME: If you reinstall your system you might already have an oserv process running. Stop oserv with the following command:

```
/etc/Tivoli/oserv.rc stop
```

Installation Sequence: Before a client can be installed, the installation of the TME server must be complete.

Encryption Levels and Passwords: During the installation process, you will have to decide on the encryption level used within your local TMR and the installation password used for further client installations or for inter-TMR connections. Refer to 2.3, "Security" on page 20 and to the *Tivoli Management Platform Planning and Installation Guide* for more information.

2.4.1.2 Planning for a UNIX Managed Node Installation

Installation Directories: By default the Tivoli Management Platform will be installed in the following filesystem areas:

- /usr/local/Tivoli

 Binaries, libraries, message catalogs, etc.

- /var/spool/Tivoli

 TME server database

These paths can be changed if required; nevertheless, for our examples we used the default paths.

Note: To save disk space, you might want to mount some directories from a file server instead of locally installing the code. Nevertheless, Tivoli recommends that you keep at least the *binaries* and the *database* local.

Required Disk Space: We recommend you install the Tivoli Management Platform into two separate filesytems, one for the binaries and the other for the database.

The initial installation of the Tivoli Management Environment needs approximately 60 MB of disk space in /usr/local/Tivoli and 8 MB in /var/spool/Tivoli.

Name Resolution: Make sure your TME server and all clients (managed nodes and PC managed nodes) are able to resolve each other's hostnames before you start the installation.

Reinstallation of the TME: If you reinstall your system you might already have an oserv process running. Stop oserv before initiating the installation process.

Installation Sequence: Before a client can be installed, the installation of the TME server must be complete.

2.4.2 Installing TME on a Server

1. Log in as root.
2. Verify the installation prerequisites as shown in 2.4.1.1, “Planning a TME Server Installation” on page 23.
3. Make sure the DISPLAY variable is set to your window.
4. Mount the binaries to install the TME:

   ```
   mkdir /tiv_code
   mount device_path /tiv_code
   ```

5. Create an installation directory and make it the current directory. The objects placed into this directory are used by the installation process and require very little disk space.

   ```
   mkdir /tiv_inst
   cd /tiv_inst
   ```

6. Start the preinstallation script:

   ```
   /tiv_code/tivoli.MgmtPlatform.3.0.rev-C.5-3-96/wpreinst.sh
   ```

 The wpreinst.sh script creates a couple of links required by the installation process. These links point from the installation directory to the installation media.

7. Install the TME:

   ```
    ./wserver -c /tiv_code
   :i1.wserver
   ```

 /tiv_code is the directory containing the code for the Tivoli Management Platform.

8. Set the installation directories and the installation options as desired. Refer to Figure 7 on page 26 for an example.

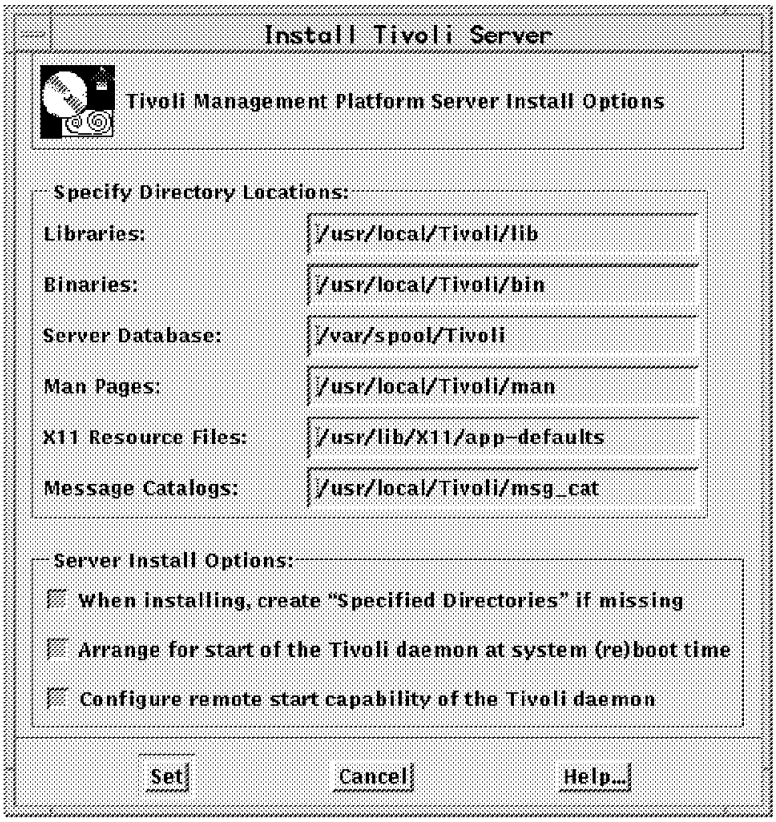


Figure 7. Installation Directories

- The Arrange for start of of the Tivoli daemon at system reboot time option creates the following entry in /etc/rc.nfs.

```
if [ -f /etc/Tivoli/oserv.rc ]; then
    /etc/Tivoli/oserv.rc start
    echo "Tivoli daemon started."
fi
```

- The Configure remote start capability of the Tivoli daemon option modifies /etc/services and /etc/inetd.conf.

```
# /etc/services
objcall         94/tcp          # Tivoli 2.0 daemon
objcall         94/udp          # Tivoli 2.0 daemon
```

```
# /etc/inetd.conf
objcall dgram udp wait root /etc/Tivoli/oserv.rc /etc/Tivoli/oserv.rc inetd
```

9. Enter the installation specific information. Refer to Figure 8 on page 27 for an example.

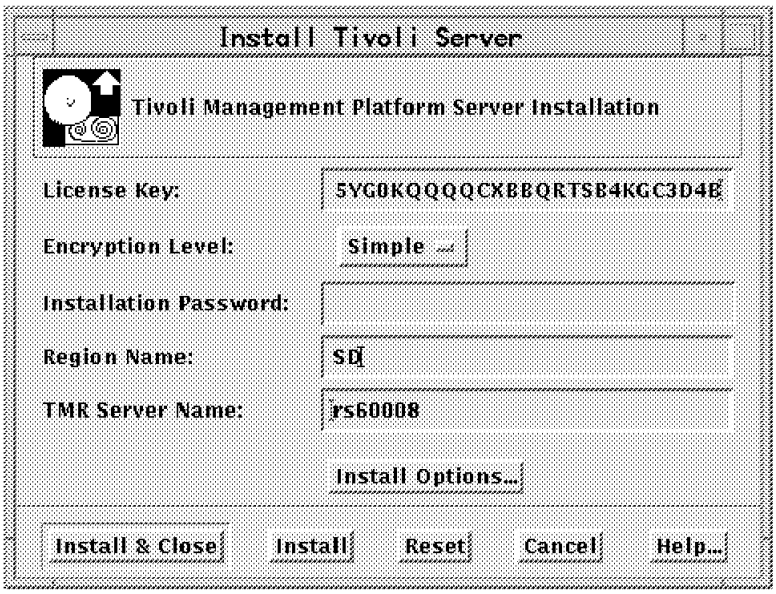


Figure 8. Installation Specific Information

- You can choose from three available encryption levels: None, Simple and DES. For more information on TME security aspects, refer to 2.3, “Security” on page 20.

 Simple encryption provides a simple key-based encryption. The impact of this encryption on performance is minimal, typically less than five percent of communication time overhead. When you select an encryption level of Simple, provide an encryption password.

 DES encryption provides a high level of security, but also has a significant performance impact, typically 12 percent or more of communication time overhead. When you select an encryption level of DES, provide an encryption password.

 If your network is physically secure, Tivoli recommends that you use Simple encryption.

 If you provide a password in the Installation Password field, it will be required for any client installation or when creating inter-region connections.

 Region Name is the initial name of your new policy region. You may change this name at any time.

10. Check the various installation messages and confirm or cancel the installation as required. After a successful installation the TME desktop will be displayed. Figure 9 on page 28 shows an example of a TME administrator desktop.

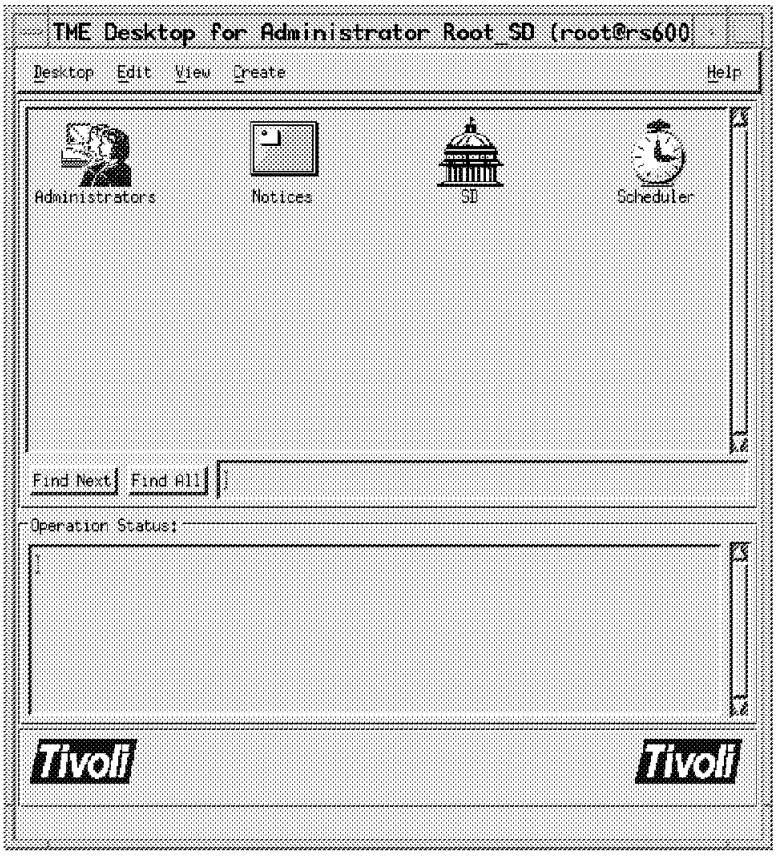


Figure 9. Initial Desktop

11. Update the administrator environment ($HOME/.profile) with the following:

```
if [ -f /etc/Tivoli/setup_env.sh ]; then
  . /etc/Tivoli/setup_env.sh
:i1./etc/Tivoli/setup_env.sh
fi
```

2.4.3 How to Start the Tivoli Management Environment

You can only start the Tivoli Management Environment if your user ID is defined as a Tivoli administrator login, and when your desktop contains at least the Notices icon.

Additionally you should update your environment with the required variables for the TME (refer to 2.4.2, “Installing TME on a Server” on page 25), and make sure that your DISPLAY variable is set correctly.

To start the Tivoli Management Environment enter `tivoli`.

Note: If you don’t get help messages, the problem might be that the message catalogs are not available. In this case export the LANG=C variable.

2.4.4 Installing UNIX Managed Nodes as TME Clients

After the Tivoli Management Platform is installed on the server, you can perform a remote TME installation on the TME clients. This is a very convenient method, since you can install all of the TME clients remotely.

1. Log in as `root`.
2. Verify the installation prerequisites as shown in 2.4.1.2, “Planning for a UNIX Managed Node Installation” on page 24.
3. Bring up the Tivoli desktop by entering `tivoli`. Refer to 2.4.3, “How to Start the Tivoli Management Environment” on page 28 for more information.
4. If you want to install TME clients into any other policy region than the initial region, make sure that the resource ManagedNode is a valid resource type in that policy region:
 - To make ManagedNode a valid resource type, double-click on the region. Then select **Properties => Select Resources** and move the resource ManagedNode from Available Resources to Current Resources. Figure 10 shows an example of the Set Managed Resources dialog.

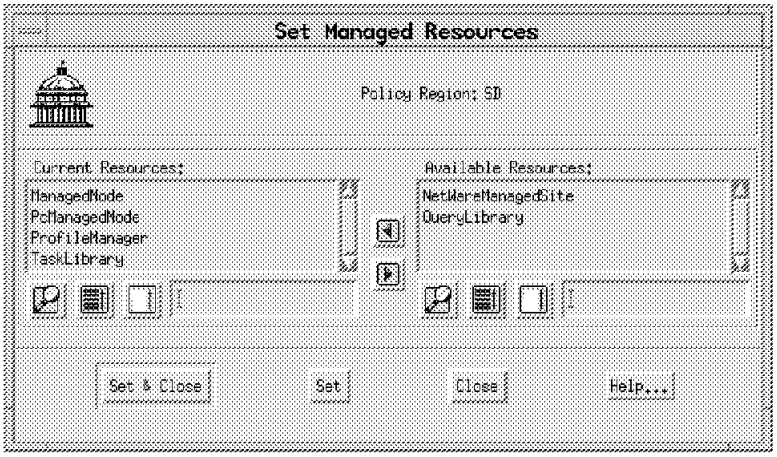


Figure 10. Set Managed Resources

5. On your desktop, double-click on the policy region in which you want to create the managed node.
6. Select **ManagedNode** from the policy region’s Create menu.
7. Define the client install options. Figure 11 on page 30 shows the Client Install dialog.

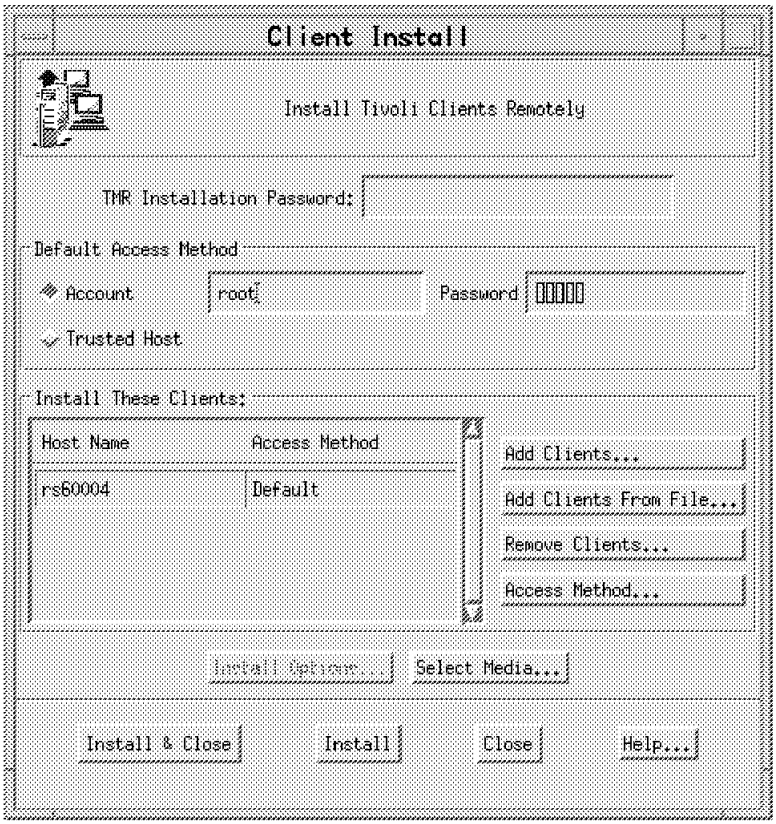


Figure 11. Create Managed Node

- If you defined an installation password when you installed the TME server, enter this same password in the TMR Installation Password field. Otherwise, leave this field blank.
- Select one of the radio buttons to specify the Default Access Method.

 Enter root if the TME server has to provide a password to interact with the managed node.

 Select **Trusted Host** if the TME server can access the client without providing a password. This means that the TME server is granted trusted host access on the client via an entry in the .rhosts file or by some other authentication mechanism.
- Select **Add Clients** to add one client at a time. Enter the clients on the Add Clients window shown in Figure 12 on page 31. Multiple clients can be entered by selecting **Add Clients From File**.

If you want to add clients from a file, the input file must have the following format:

```
rs60004
rs60005,
rs60006,password
```

hostname only means you want to add rs60004, for example, and use the Default Access Method.

hostname followed by a comma means you want to add rs60005, for example, and use Trusted Host Access.

hostname followed by a comma and a password means you want to add rs60006, for example, and enter root to use a password for authentication.

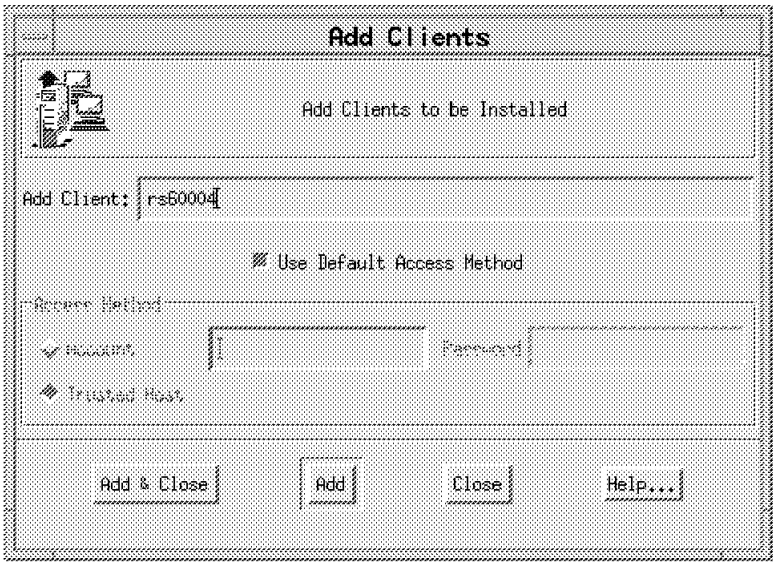


Figure 12. Add Client Dialog

8. Activate **Select Media** and select the path to your Tivoli Management Platform installation media. In our environment, we used

 `/tiv_code/tivoli.MgmtPlatform.3.0.rev-C.5-3-96.`

9. Select **Install & Close** to start the client installation.

Note: Now that the Tivoli Management Platform is installed, you might want to install the newest *Service Pack* (Patch). See 2.4.6, “TME Service Pack Installation” on page 40 for an example.

2.4.5 Installing PC Managed Node as TME Client

Once the TME Server is installed, you may install TME clients. The installation of TME PC managed clients is different from the installation of UNIX managed nodes, so you will not be able to install them remotely from the TME server. You have to install them by hand.

The TME includes two different types of agents: an IP agent and an IPX/SPX agent. The IP agent can be installed on a PC running Windows 95, Windows NT, Windows, DOS, NetWare, or OS/2. In our project we installed Windows NT, Windows 3.X, and Windows 95. The following scenario will describe the installation of Tivoli on a node running Windows 95. Insert the Tivoli Management Platform for NT CD-ROM in your CD drive on the PC.

1. Click on the **My Computer** icon. The pop-up window will show your devices defined at the PC. Double-click on the **CD-ROM** icon.

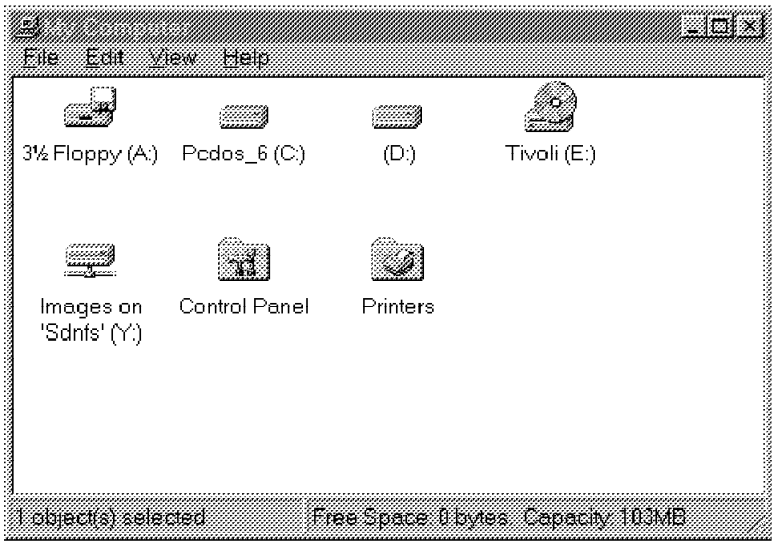


Figure 13. Devices Defined at Your PC

In our case the CD-ROM is drive E.

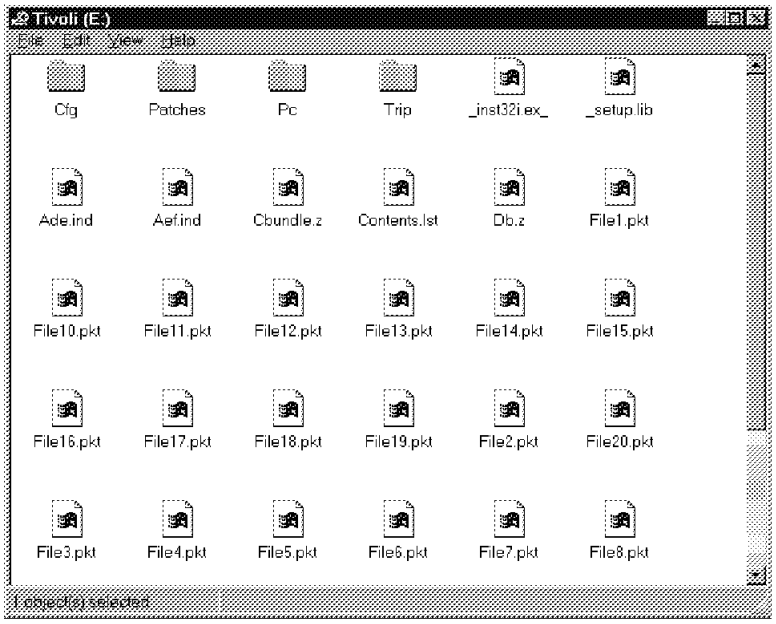


Figure 14. Files and Directories on the CD-ROM

2. As shown in Figure 14 you may see the Pc directory, which contains the software for TME clients. Get to the Setup program by double-clicking on **Pc**, **TCPagent**, and **Cd**. Now you get the window shown in Figure 15 on page 33.

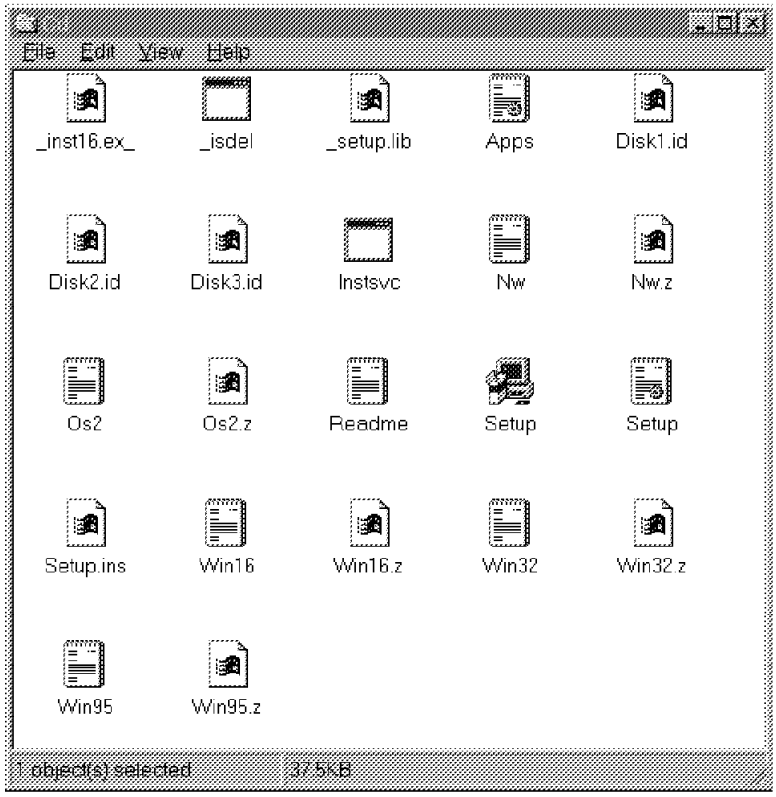


Figure 15. Directory of TCP/IP Client Software

3. Selecting the **Setup** program will start up the Tivoli Systems TME Agent Setup.

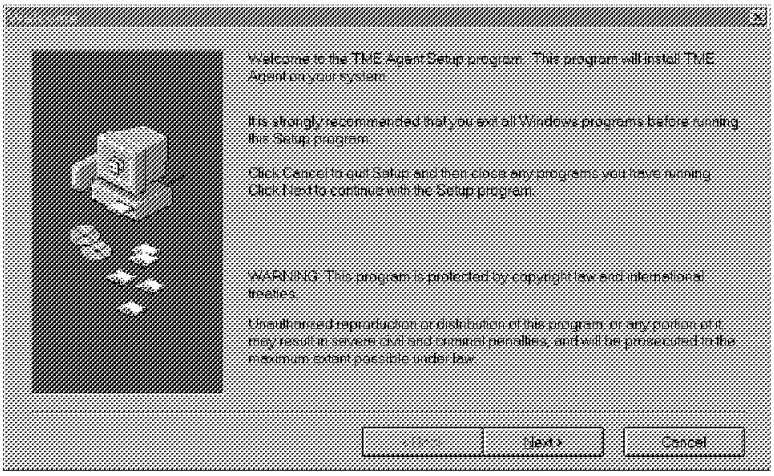


Figure 16. Welcome Panel

4. Select the **Next >** radio button to continue the installation.

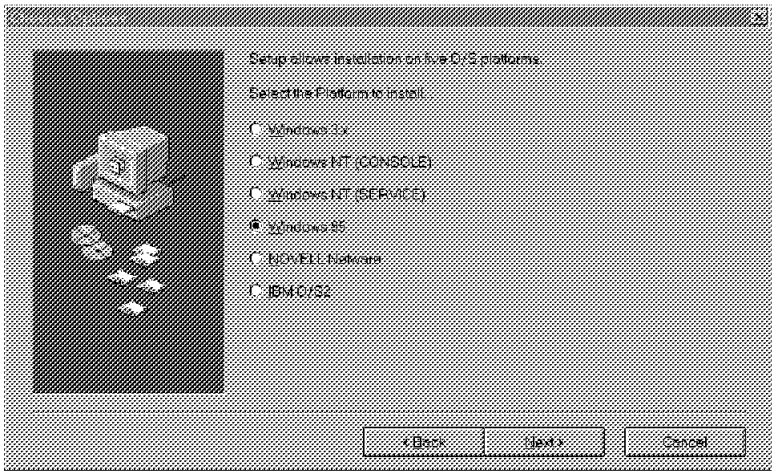


Figure 17. Platforms

5. Choose the platform on which to install the PC client. If you want install the agent software on Windows NT, there will be two agents available. Select the service agent if you want the agent to run continually when Windows NT is running. Select the console agent if you want the agent to run only when the user is logged in. That is, the console agent runs only if it is specified in the Startup group or if the user explicitly starts it. Select the **Next >** button to get Figure 18.

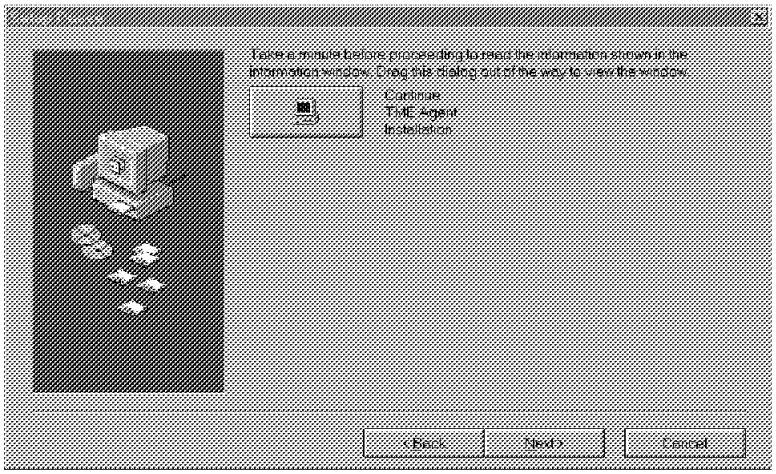


Figure 18. Confirm Continuing Installation

6. Select **Next >** or **Continue TME Agent Installation**.

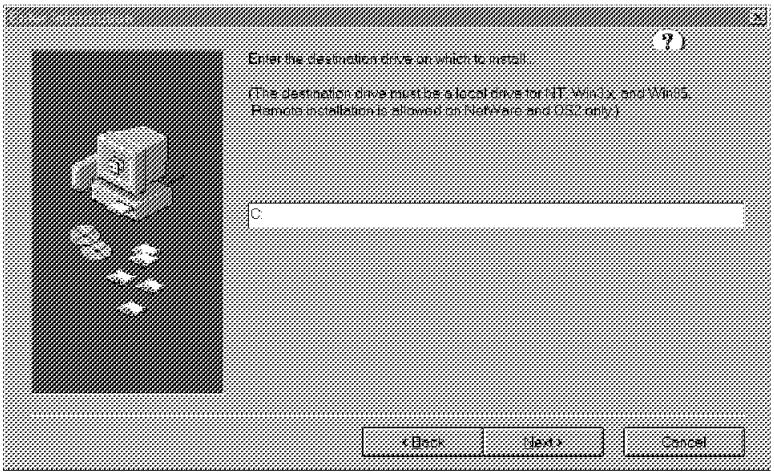


Figure 19. Enter Destination Drive

7. Enter your choice of destination drive on which the TME agent should be installed at your PC. The default installation device will be C. Select **Next >** to continue the installation.

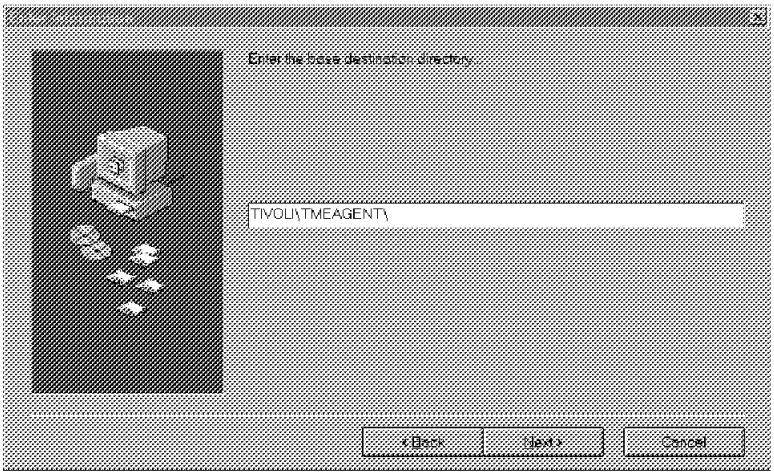


Figure 20. Enter Destination Directory

8. Enter your destination directory where the agent software should be installed at your PC. The default is the \TIVOLI\TMEAGENT\ directory. Depending on the platform you chose, the SETUP program copies the files to the machine and appends the platform name to the directory name. For that information, refer to the *Management Platform Planning and Installation Guide*.

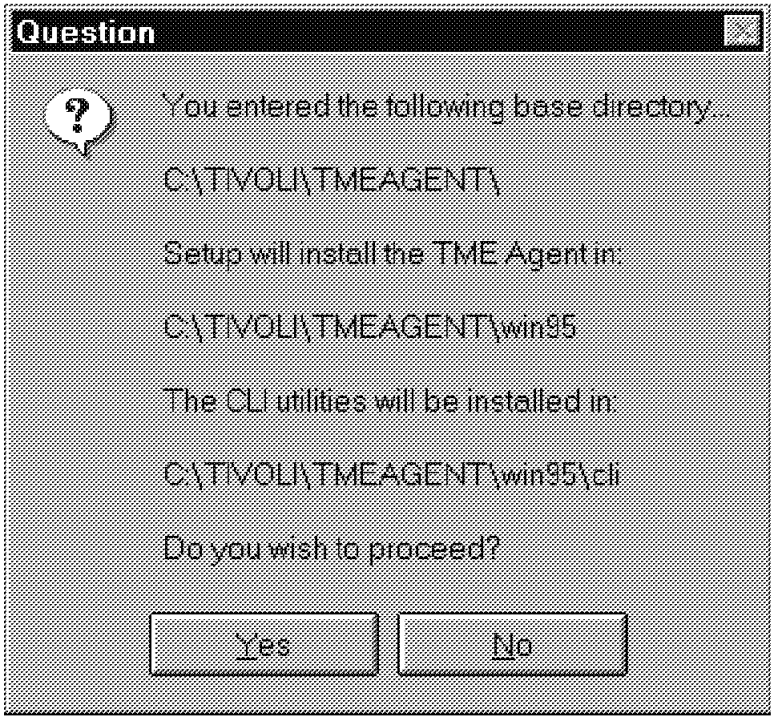


Figure 21. Accept Installation Settings

9. The setup program prompts you to accept your choice of settings. If you want to accept the settings, select the **Yes** radio button. If not, select the **No** radio button and you have the option to change the default installation directory and default drive settings.

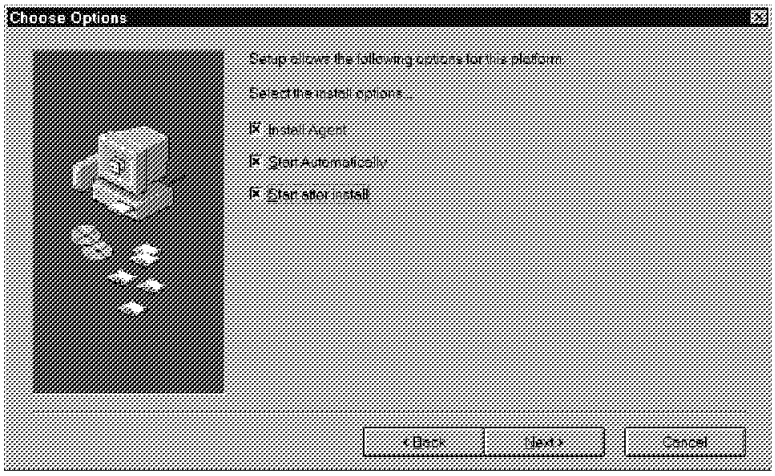


Figure 22. Select Startup Functions

10. Select the startup options of the TME agent.

 - Install Agent

 This is the check box to copy agent's binaries to the hard disk. Select this option if the TME Agent was not installed previously.

 - Start Automatically

Select this check box if the PC agent should start up every time the system starts. For modification of startup files, refer to *TME Management Platform Planning and Installation Guide*.

- Start after install

 Select this check box if the agent should start up after installation.

Select **Next >** to continue the installation.

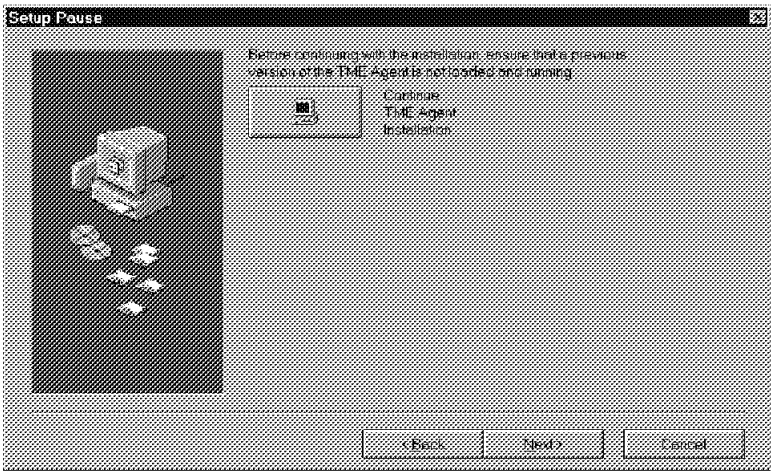


Figure 23. Confirm Continuing Installation

11. Select **Next >** or **Continue TME Agent Installation**.

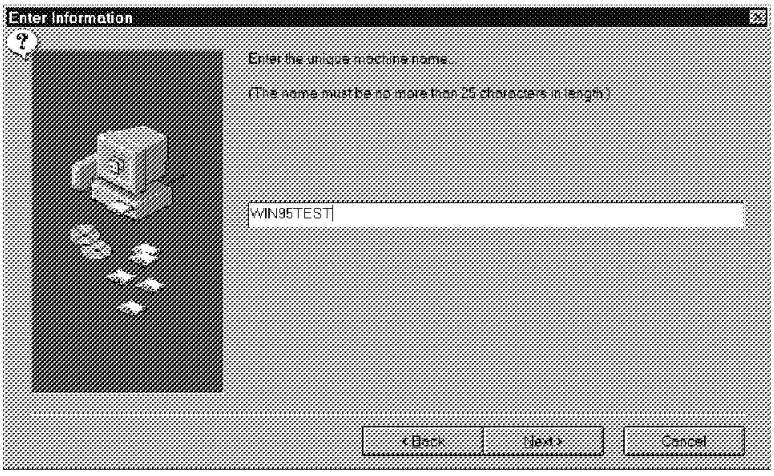


Figure 24. Enter Name of the Machine

12. Enter the name of the machine where the PC agent should be installed. For more information, refer to *TME Management Platform Planning and Installation Guide*. Select **Next >** to continue the installation.

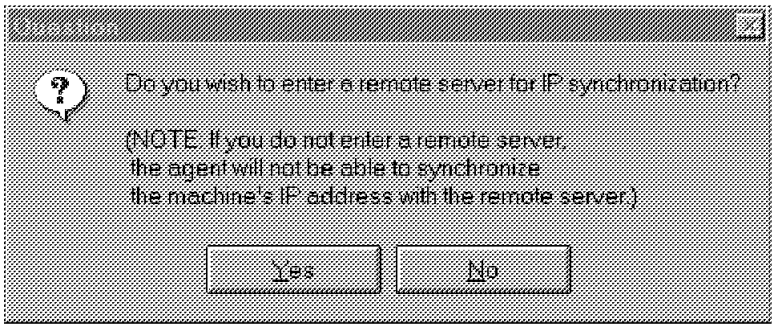


Figure 25. Remote Server for IP Synchronization

13. Confirm if you want the agent to synchronize its IP address with the server by selecting **Yes**.

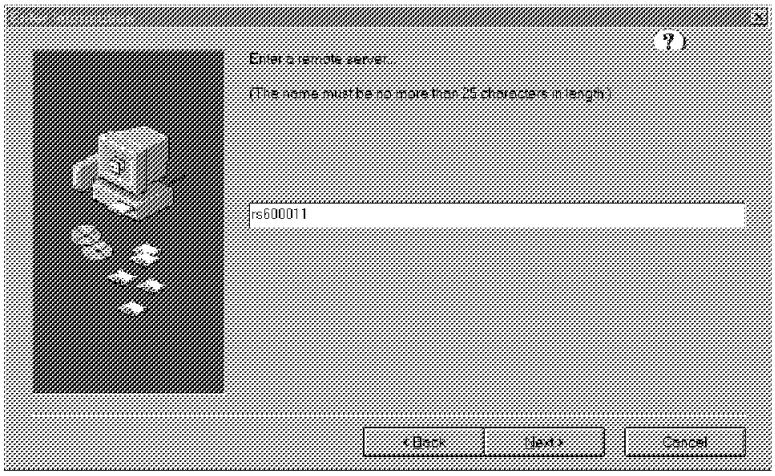


Figure 26. Remote Server Name

14. Enter the name of your remote server and select **Next >**. In our project we used rs600011.

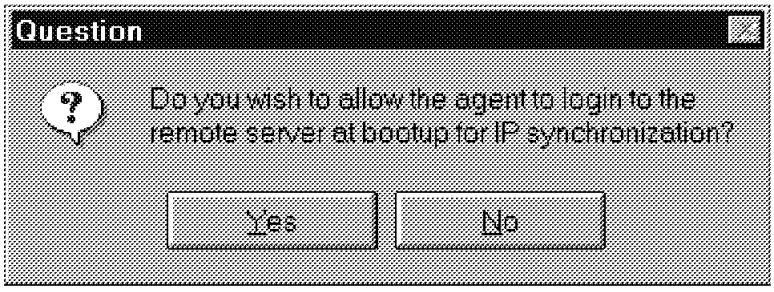


Figure 27. Allow IP Synchronization at Bootup

15. To give the agent access for synchronizing its IP address at bootup select **Yes**.

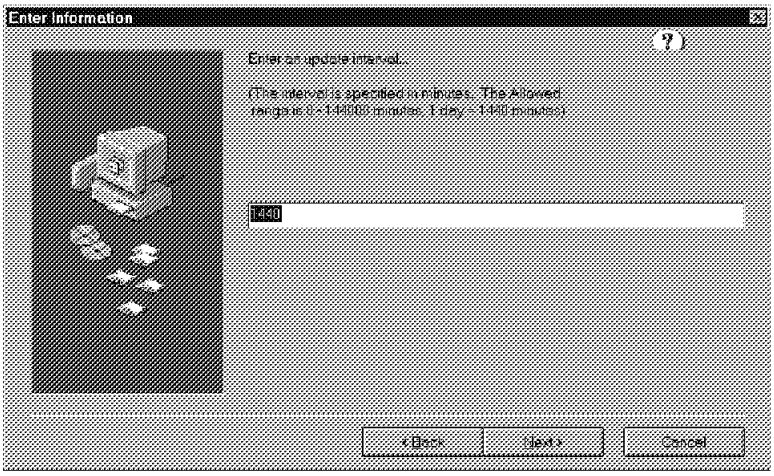


Figure 28. Enter Update Interval

16. You have to specify how often the agent will be able to synchronize with the server. Enter your choice in minutes. Select the **Next >** button to continue the installation.

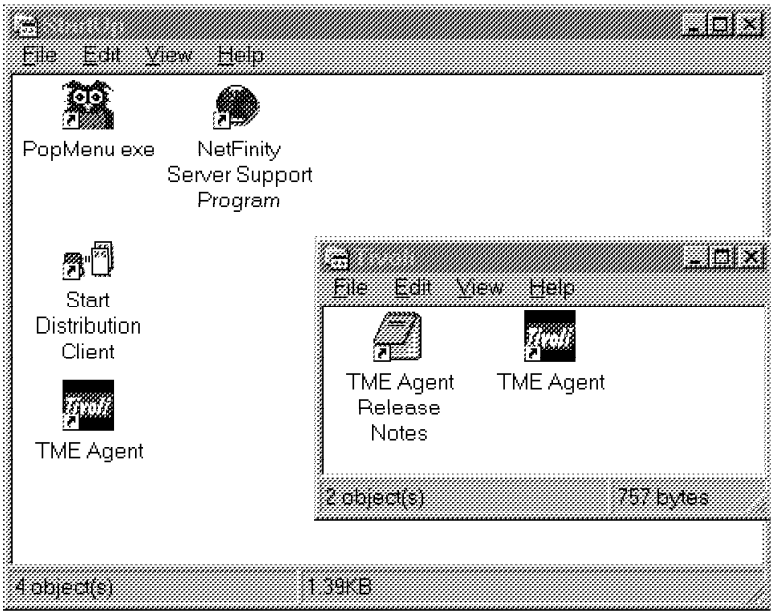


Figure 29. Installation Completed

17. The setup program installs the agent based on your selection at the startup group. A group called Tivoli will be added to your desktop including the Tivoli agent. Refer to Figure 22 on page 36. We selected all options so that setup installs the agent in the startup window, too.

2.4.5.1 Updating the TME Agent on PC Managed Nodes

Since the PC agent software is installed successfully to the PC, you may think you can install updates from TME server, but you can′t.

NOTE

To update a PC managed node, install a new TME agent.

2.4.6 TME Service Pack Installation

In the following example we install the Tivoli Management Platform 3.0 Service Pack 01. It is located in the /TIV_MGMT_PLATFORM_3.0.1 directory on the installation media.

1. Log in as `root` or as any other TME administrator with the `super` or `install_product` role.
2. From the administrator desktop select **Desktop**. Then select **Install = > Install Patch**.

 Note: There might be an error message, saying that there is no PATCHES.LST file. Disregard it and select the installation directory from either the File Browser or the Install Patch window.
3. On the Install Patch window (shown in Figure 30) select **Select Media**, then the directory containing the service pack, and finally **Set Media & Close**.
4. Select **Patch to Install** and **Clients to Install On** and start the installation.

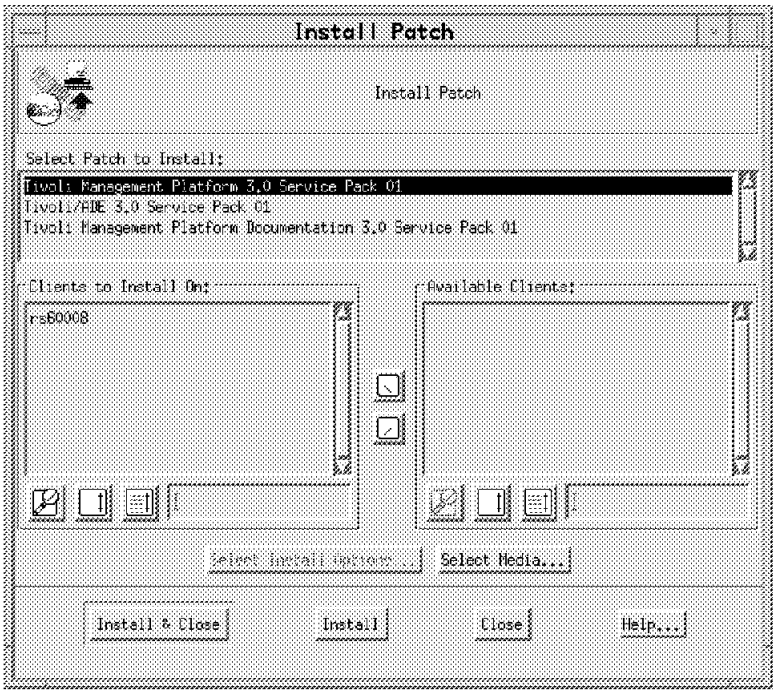


Figure 30. Patch Installation

2.5 Product Installation

The Tivoli core applications (Tivoli/Courier, Tivoli/Admin, etc.) are normally installed remotely, either from the Tivoli desktop or the command line.

To install the products you need the following:

- The product install code
- License Key (if applicable, this is not required for TME 3.0 or later)
- Installation password for the TMR

- An administrator ID with either super or install-product authorization roles for the TMR

 You need to start the TME Desktop with this admin ID.

2.5.1 Product Installation Using the TME Desktop

1. Select **Desktop** from the TME desktop menu bar.
2. Select **Install = > Install Product** from the pull-down menu. This will display the Install Product dialog as shown in Figure 31.

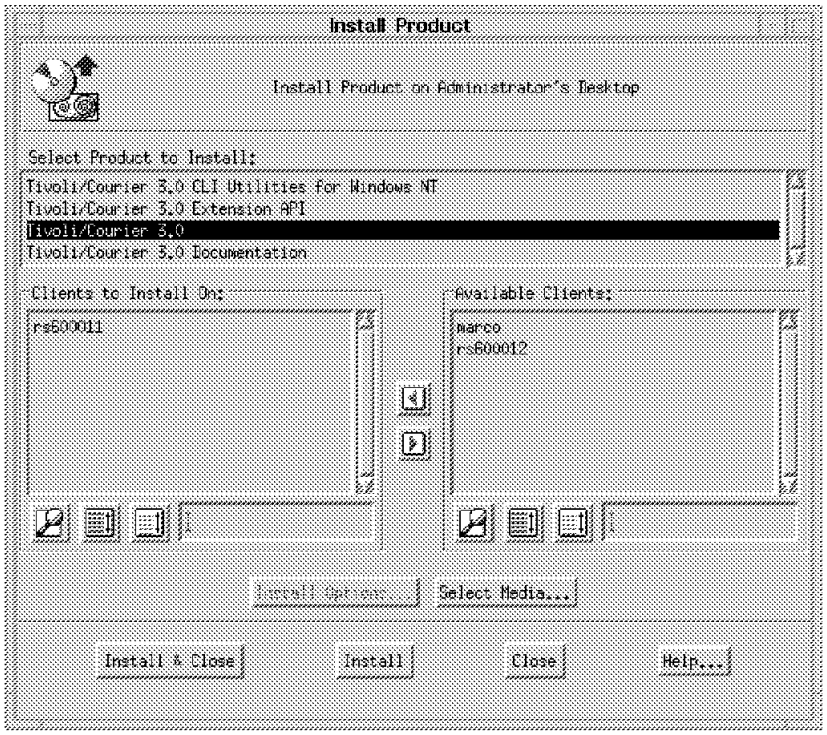


Figure 31. Install Product Dialog

If the Select Product to Install list does not have the product you want to install, or it is empty, then click on **Select Media** in the Install Product dialog to change the path to the install media. This will display the File Browser dialog as shown in Figure 32 on page 42.

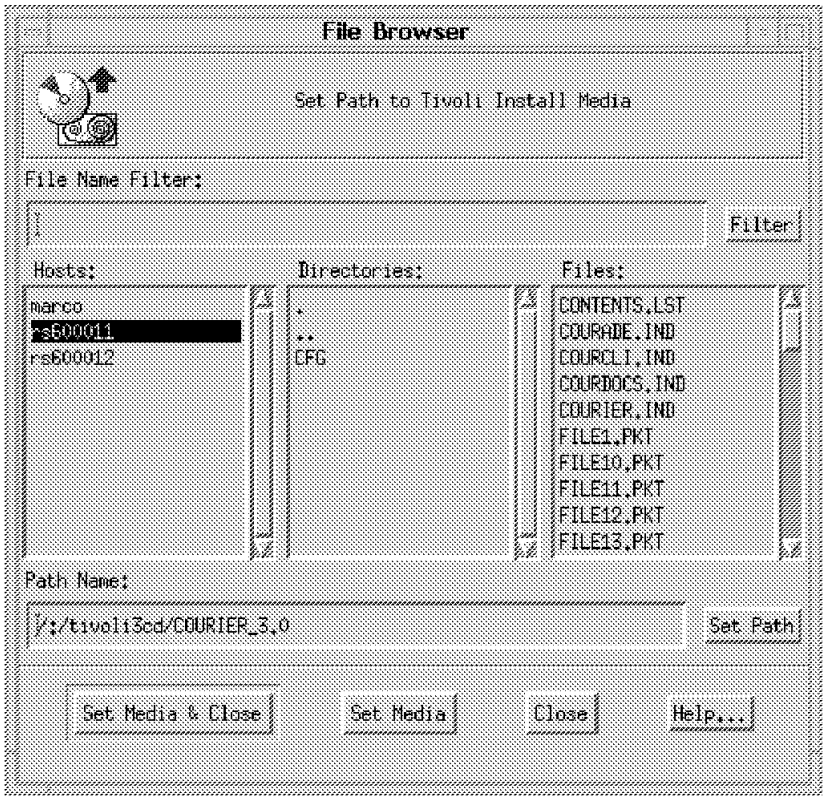


Figure 32. File Browser Dialog

Use the File Browser dialog to change the path to the location where you have the installation code. You can install the product from the CD-ROM or copy it to a disk file system and install it from there.

3. Highlight the product by clicking on it in the Select Product to Install list.

 You can only install one item at a time from the list. If you need to install another item, select this item again after you finish installing the first one.

 When you have selected the product in the list, you may be prompted to enter some additional installation options. These are options that are specific to the product in question, such as special disk directories or configuration parameters. Usually you can safely leave these to default.

4. Fill in the Clients to Install On list by selecting clients from the Available Clients list (the two lists combined will display all of the valid clients available in the TMR, including the TMR server).

 Note that once the TME platform is installed, additional products can be installed on any machine in the TMR, thereby allowing you to distribute application load.

5. Click on **Install & Close** to install the product.

 This will close the Install Product dialog when the installation is completed. If you need to install another product, then click on the **Install** button instead. This will keep the Install Product dialog open at the end of current product installation.

6. When you select either **Install & Close** or **Install**, you will get a Product Install dialog requesting your confirmation to continue.

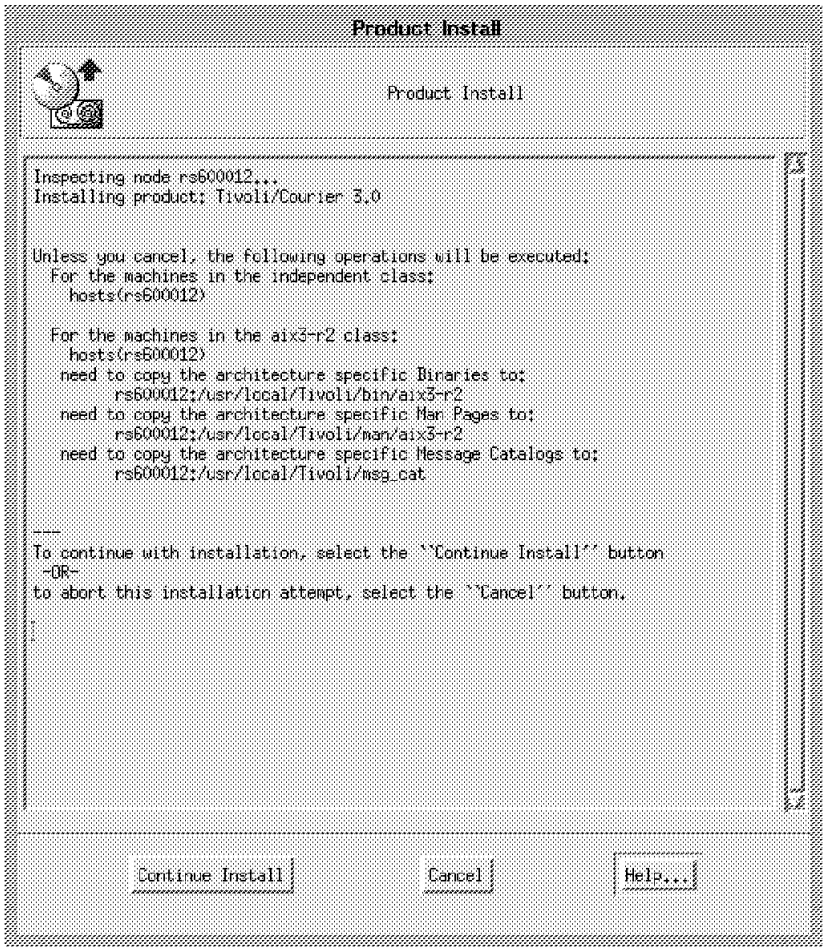


Figure 33. Product Install Dialog

The dialog lists the actions that the install program is about to execute. It may also give you error messages. If there are no error messages, click on **Continue Install** to continue.

The Product Install dialog will display status information during the product installation. At the end of installation it will display a dialog as shown in Figure 34 on page 44.

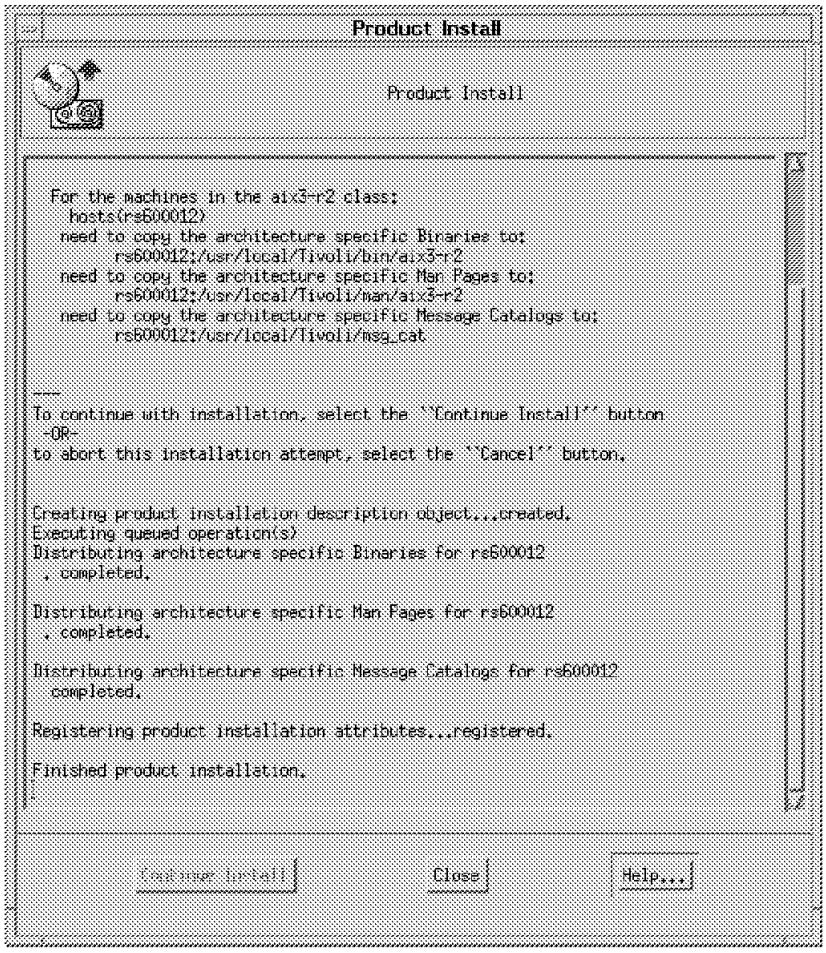


Figure 34. Product Install Dialog at the End of Installation

2.6 Diagnosing Installation Problems

Usually, if you follow the instructions in the product documentation, installation of TME components is straightforward. However, we set up a number of different scenarios for our project and in doing so we came across a few installation problems. The cause for failure was not apparent in every case. For that reason you may find the following descriptions of the problems we encountered to be a useful source of clues if you have similar problem symptoms.

2.6.1 UNIX Environment Problems

We received the following message when we entered the wserver command. We were using telnet to access the system.

```
using the interpreter type aix4-r1 as set in your environment.
wserver: Your $DISPLAY environment variable is not set, and you did not
select a non GUI install by setting the environment variable $DOGUI to
something other than $DISPLAY.  To recap:

If you want to use a GUI (X11) based install, set $DISPLAY and unset
$DOGUI

If you want to use a CLI (command line) based install, set $DOGUI=no
```

```
in this case, you can still have your $DISPLAY set to bring up
the initial Tivoli desktop.
```

This means that, assuming that you want to use the GUI installation option, you should set the DISPLAY environment variable. In Korn shell, it is set by entering `export DISPLAY=myhost:0`. You may then see a message like the following:

```
Xlib:  connection to "rs600010:0.0" refused by server
Xlib:  Client is not authorized to connect to Server
Error: Can't open display: rs600010:0
```

This means that you need to give the server machine X-windows access on your own machine. The command to do this is `xhost`. If you are not too concerned about security, the easiest entry is `xhost +`, which gives access to all machines.

Once the installation is complete, you normally set up the Tivoli environment by running the following command:

```
. /etc/Tivoli/setup_env.sh
```

We showed an example of adding this to the .profile script on 28.

One related problem we encountered was where the environment variables set by this script had already been hardcoded into root's .profile, pointing at an old release of the Tivoli code. This caused a number of problems with the installation. This is unlikely to happen in other environments. However, similar problems are possible. For example, some other product or application may use the $BINDIR, $LIBDIR or $DBDIR environment variables for its own purposes. If you get unexplainable errors, check that you are not suffering from such a conflict.

2.6.2 Space Problems

On a number of occasions we had installation failures due to not having allocated enough space in the Tivoli filesystems. The installation processes attempt to prevent you from continuing by presenting the message shown in Figure 35.

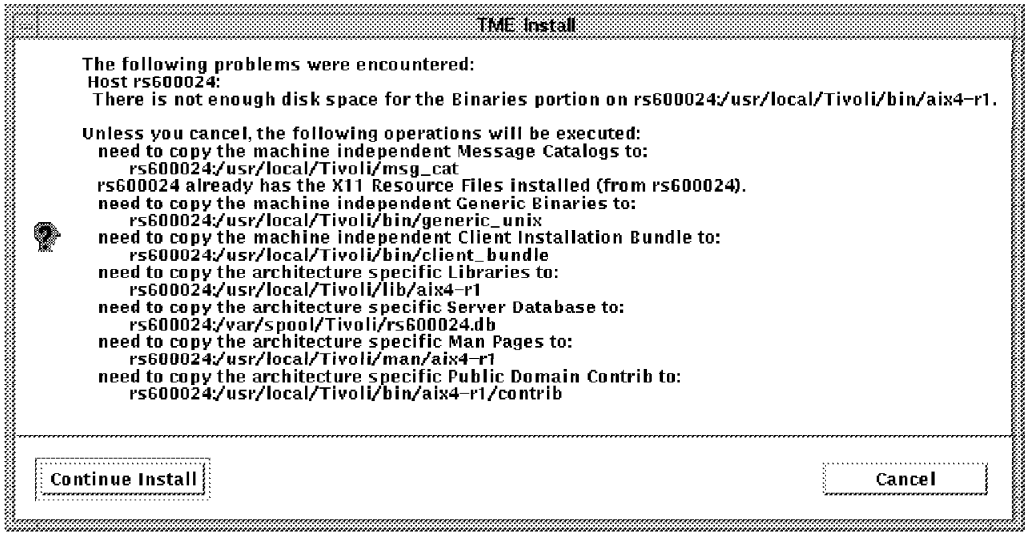


Figure 35. Not Enough Space to Install Message

To fix this, simply extend your existing filesystems or create a new filesystem using the guidelines for size specified in the installation guide.

Unfortunately the error message is only one message out of many on the window. If you disregard it and select **Continue Install**, the installation continues even with the errors. In that case you may get the following messages:



Figure 36. Continuing without Enough File System Space

If you see this message, the df command may not show the filesystem to be full. For example, in our installation the df showed the following:

```
/dev/lv06        356352    64332   81%     6106      6% /usr/local/Tivoli
```

However, if you see error messages like the one at the top of Figure 36 showing that a copy function has failed, it is a good indication that you have run out of space somewhere.

If you have not allocated enough space for your server database, you will see the messages shown in Figure 37 on page 47.

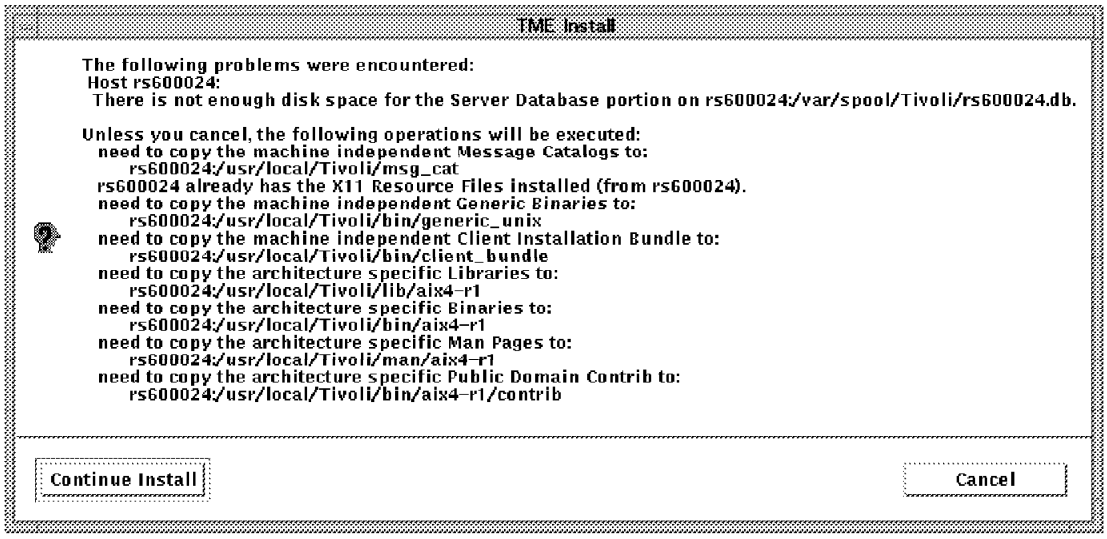


Figure 37. Insufficient Space for the Server Database

In this case, if you miss the message and select **Continue Install**, the installation may appear to complete successfully (see Figure 38 on page 48).

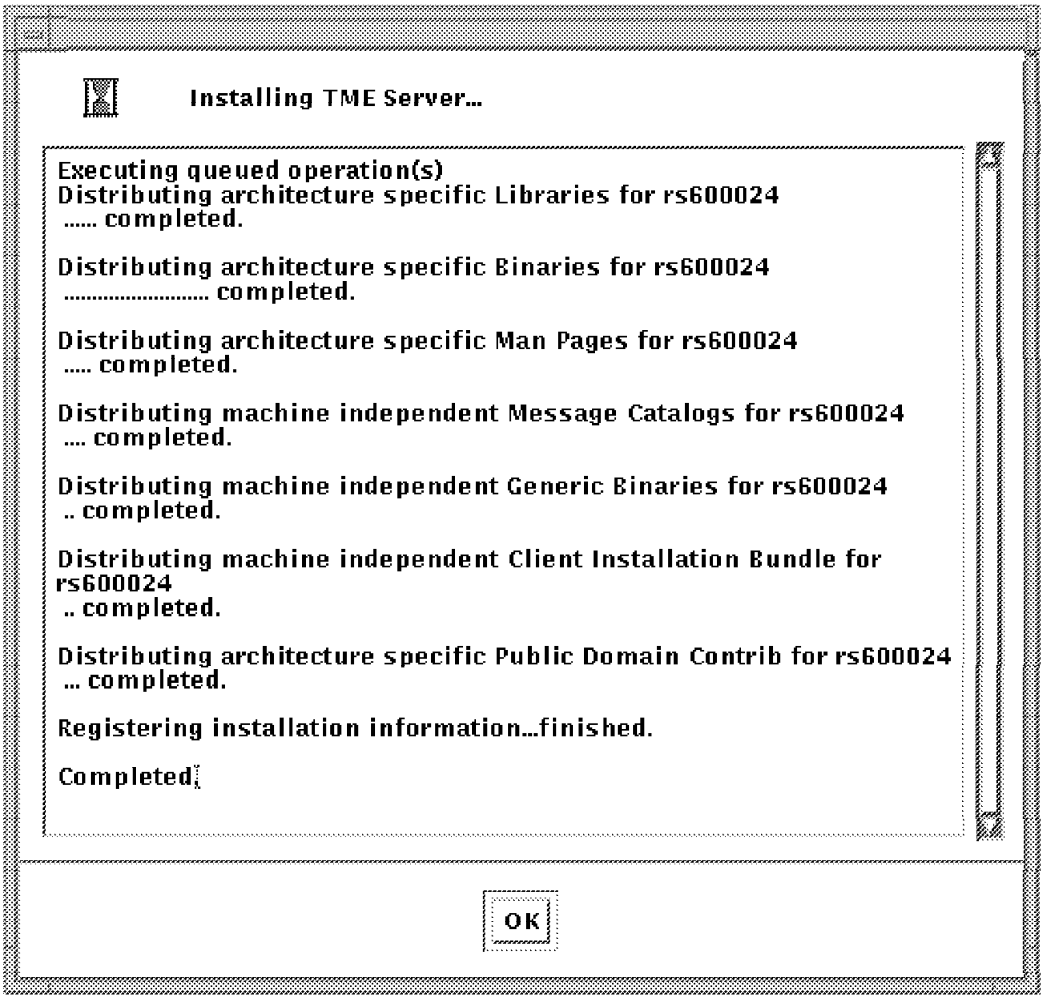


Figure 38. Seemingly Successful Installation Messages

As before, you may find that the df command does not indicate a full filesystem, for example:

```
/dev/lv01        4096       1868    55%       19      2% /var/spool/Tivoli
```

However, you will probably find that the oserv daemon is not running and that trying to start it manually with the command `oserv -k /var/spool/Tivoli/rs600024.db` produces the following message:

```
get_boot_info failed (30)
!oserv: odlist init failed. requested resource not found. (30)
```

The solution in this case was to increase the space allocated to the server database, remove the files currently under /var/spool/Tivoli and reinstall.

Starting oserv

Note that this is *not* the normal way to start oserv. The preferred command to use is `odadmin start`. However, in this case starting oserv directly allowed us to see the error message, instead of having to look for it in $DBDIR/oservlog.

2.6.3 Reinstallation

Because of the fluid nature of our lab environment, we had to reinstall both the TME server and the clients a number of times. In general, to reinstall the server you have to erase the contents of the Tivoli directories and restart the original installation procedure.

You may not always have to remove everything before performing a reinstallation. For example, if you have a space problem on the database directory and the bin, lib, msg_cat and man files appear to install correctly, it is not necessary to remove those directories before reinstalling. However, at this stage, you will be losing nothing by removing all of the files. If you have any doubts about the status of the installation, it is best to remove everything. Remember that you can also place an exclamation point (!) after each directory name in the install options window (see Figure 7 on page 26) to force a reinstallation of those files.

If you choose not to remove the files, you will see messages indicating what is already installed and what now needs to be copied (see Figure 39).

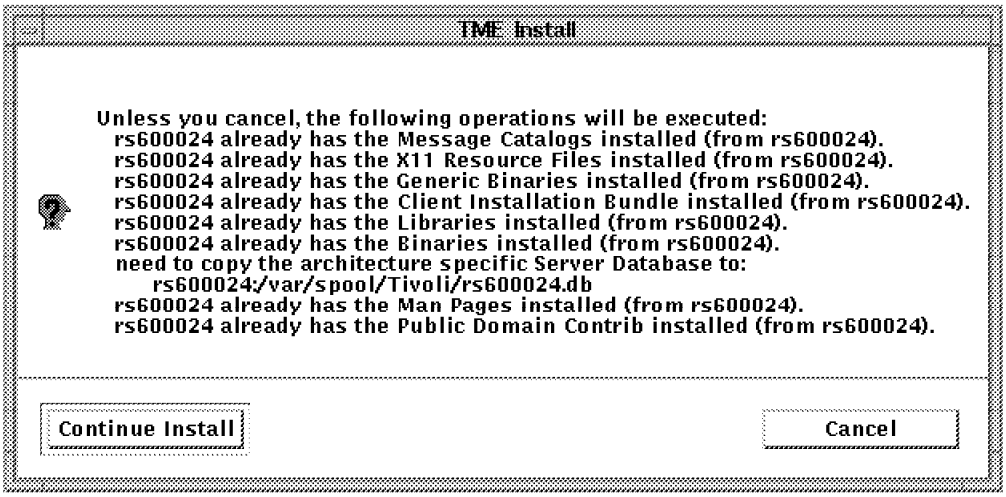


Figure 39. Messages When Installing with Some Files Already Installed

2.6.4 Problems Installing Products and Patches

After installing the server, you may next want to install some additional products or patches (as described in 2.4.6, “TME Service Pack Installation” on page 40). The first time the Install Product or Install Patch option is selected after Tivoli is started, you will see the warning shown in Figure 40 on page 50.

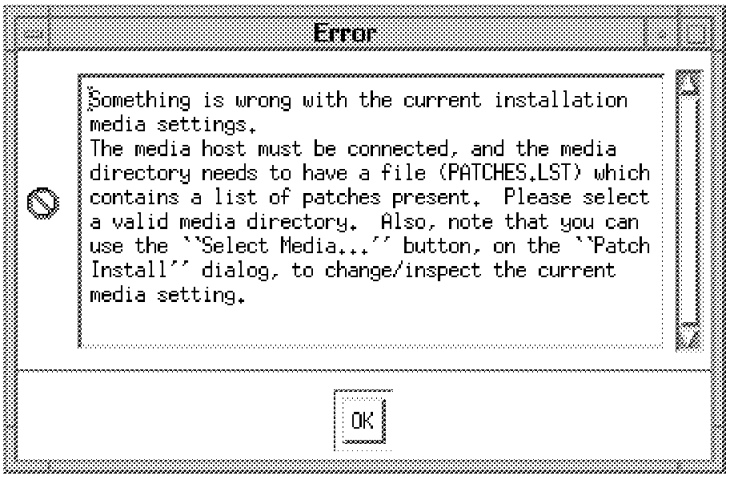


Figure 40. Warning on First Product Install

This warning looks serious, but in fact it only says that you have not yet defined a source directory. In addition, the File Browser window mentioned in the error message will appear automatically after you click on **OK**. It may take several seconds to appear, so do not try to select Install Product from the menu again, or else you will have multiple File Browser panels on your window.

2.6.5 TME Client Installation Problems

The TME base code is quite large, so one recommended option to conserve disk space is to use NFS to mount some of the file systems from the TME server. The installation guide suggests that the libraries and database should be stored locally. We decided to also store the X11 resource files locally. The remaining directories are bin, man and msg_cat. We exported these directories from our server and created NFS mounts on our client.

When we then tried to create the client managed node, we received the errors shown in Figure 41 on page 51.

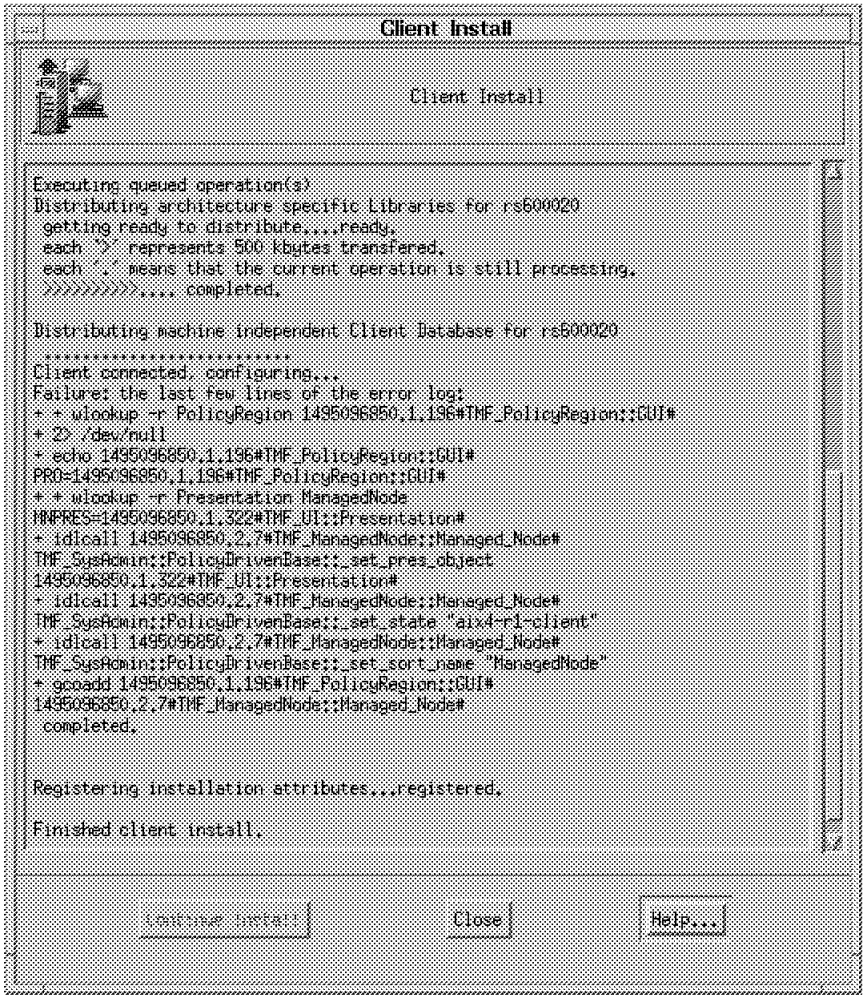


Figure 41. Error Messages from Client Install

We found that the error log mentioned in the message is /tmp/client.cfg.error. However, it gave us no clues as to where the problem was, except that the error appeared to occur as the installation process executed the gcoadd command. We found at this point that the oserv daemon was running successfully on the client. But when we closed the Client Install window, we did not see a new Managed Node icon on the Policy Region window.

We then used the following command to check if the new Managed Node actually existed in the oserv database on the server:

```
wlookup -a -r ManagedNode

rs600020        1495096850.2.7#TMF_ManagedNode::Managed_Node#
rs600024        1495096850.1.323#TMF_ManagedNode::Managed_Node#
```

From this display you can see that two managed nodes are known: the server (rs600024) and our new client (rs600020). This seemed to suggest that rs600020 had been registered correctly in the database, but it still did not appear on the Policy Region display.

Our next step was to check the integrity of the oserv database with the wchkdb command. This command returned the following errors (among various other inconsistencies):

```
Unable to examine or fix member "rs600020" (1495096850.2.9#TMF_
ManagedNode:Managed_Node#). The following error was received while
attempting to access the member:
->      System Exception: communication failure: completion status: NO
        destination dispatcher unavailable
```

This gave us the clue we needed. The message says that oserv on the server is unable to communicate with oserv on rs600020. We know that oserv was running on both machines, but for some reason they would not communicate. We surmised (and subsequently proved) that the problem lay in our use of NFS and the levels of code being shared with the server:

- TME Management Platform on the server was at Level 3.0 with a 3.0.1 patch installed on top.
- The TME Management Platform being installed on the client (rs600020) was the base level 3.0.

Therefore, the client had access to the Level 3.0.1 binaries through the NFS mount, but its library files were at Level 3.0. This mismatch caused oserv to behave incorrectly.

The lesson to be learned here is to be very careful with how you use NFS so as not to cause incompatibilities in levels of code. In fact, unless disk space is very limited, our experience suggested that installing all of the files locally is safer and more reliable than using NFS.

2.6.6 Some Useful Diagnostic Commands

Before running any Tivoli commands from the command line, remember to set up the environment if this has not already been done:

```
. /etc/Tivoli/setup_env.sh
```

We found the following commands useful when diagnosing installation problems:

wlookup -R To list resource types in the database

wlookup -r <resourcetype> -a To list resouces of a specific resource type

wchkdb To check the database for inconsistencies

wchkdb -u To fix database inconsistencies

Useful information is found in the following log files:

- /var/spool/Tivoli/rs600024.db/oservlog
- /tmp/*inst* (these may not have Tivoli in their names)

2.7 Setting up a Backup Schedule

It is important to take regular backups of the Tivoli object database. The first reason for taking backups is a common sense one: the TME database contains important systems management information that you want to protect from hardware or software failures.

The second reason for taking backups is less obvious. When you install TME products, the installation process is in fact performing a sequence of actions:

- Installing code on the managed node(s)
- Executing local configuration programs

- Updating configuration entries in the TME database

If there is some problem during installation, or if you need to reinstall a product, the easiest way to reverse the process is to restore the TME database from a backup taken prior to the installation.

There is a built-in facility for scheduling a backup as a regular job and we recommend that you set this up after the initial TME installation.

2.7.1 Backup Prerequisites

At first, you have to decide the following:

- When you will take backups
- Where you will store backups

Also, before creating the backup schedule, you need to confirm:

- If administrators who want to schedule backups have admin and backup authority roles over the TMR
- If administrators have write permission to the target backup directory
- If administrators are a member of the Scheduler notice group

If the administrators don't meet the above criteria, you need to set them. Also, you need enough free disk space for backups. You can confirm the current free disk space on the system by entering the `df` command. You can estimate the required disk space for one-time backup from TME GUI:

- Select **Desktop...** from the desktop menu bar.
- Select **Backup...** from the pop-up menu.
- Select at least one managed node that you want to backup.
- Select **Estimate Backup Size** button.

2.7.2 Example Backup Schedule

This example will be done with the following conditions:

- Backups will be taken under /var/spool/Tivoli/backups (which is a default directory for the backup) on TMR server rs600011 in prod-region.
- Backups will be taken at 3:00 a.m. on every weekday.
- The scheduling administrator user ID is root.

 To schedule a backup, perform the following steps:

 1. Start the TME desktop with the `tivoli` command.
 2. Select **Desktop** from the menu bar, then select **Backup...**. You will see the Backup Tivoli Management Region dialog.
 3. Select all nodes from **Available managed nodes:**, then add them to **Backup these managed nodes:** (see Figure 42 on page 54).
 4. Change **Device/File:** to /var/spool/Tivoli/Backups/all_nodes_DB_%t. In this case `%` means date, and `t` means time. So, for example, a backup taken on August 13 at 13:00 will have the file name all_nodes_DB_Aug13-1300.

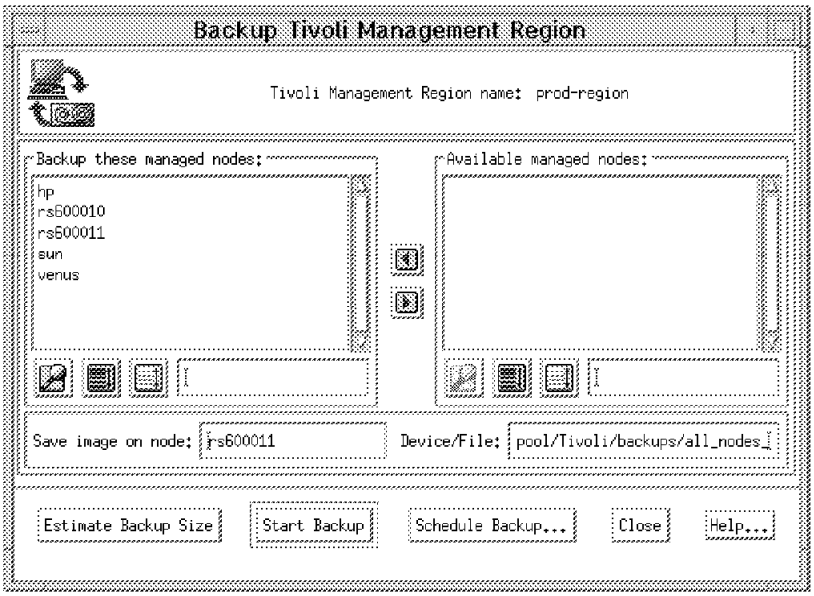


Figure 42. Backup Tivoli Management Region Dialog

5. Click on **Schedule Backup...**. The Add Scheduled Job window will appear (see Figure 45 on page 56).
6. Give the backup job a label and change the hour in the Schedule Job For: field to 3. Select **Repeat the job indefinitely** in the Repeat The Job: field. You have to set an interval time when you schedule a job repeatedly. Set the The job should start every field to 24 hours to take a backup every day.
7. To receive the notification of the scheduled backup results from the scheduler, select **Post Tivoli Notice:** in the **When Job Complete:** option, and click on **Available Groups...**. You will see the Select Notice Groups panel shown in Figure 43.

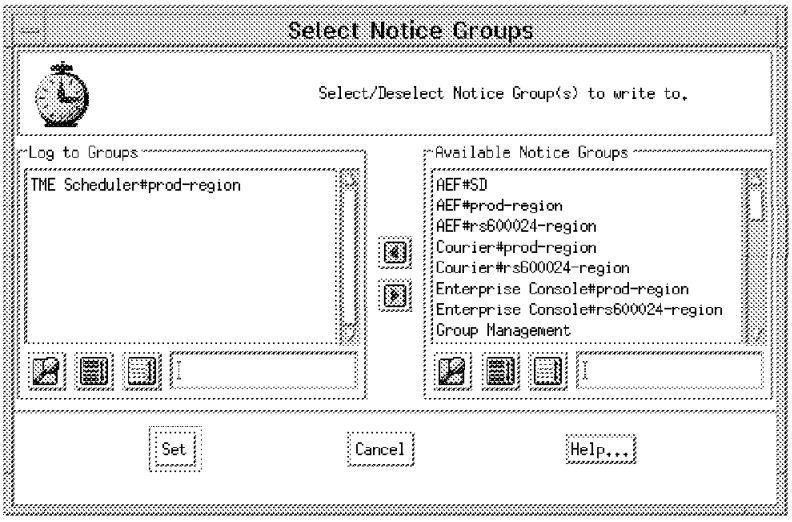


Figure 43. Select Notice Groups Panel

8. Select **TME Scheduler#prod-region** in the Log to Groups field and click on **Set**.
9. On the Add Scheduled Job panel, select **Set Retry/Cancel Options...**. The Set Retry/Cancel Options dialog appears (see Figure 44 on page 55).

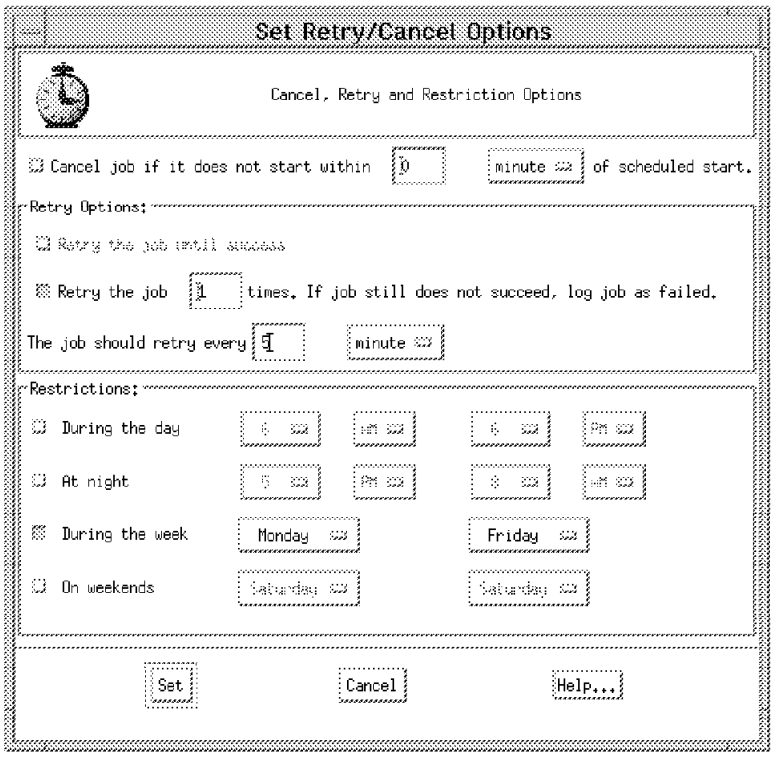


Figure 44. Set Retry/Cancel Options Panel

10. Select **Retry the job 0 times...** from the Retry Options: field, and enter 1 to replace the 0. Also enter 5 to replace the 0 in the **The job should retry every 0 minutes** field.
11. Select **During the week** in the Restrictions: field, and set Monday as the first day of the week. By this, DB backup will be taken from Monday to Saturday at 3:00 a.m. on every week. Click on **Set**.

 Note: When you want to take backups from Tuesday at 3:00 AM to Saturday at 3:00 AM, you should not select both During the week and On weekends on the same panel but should create two schedules for During the week (from Tuesday to Friday) and On weekends (from Saturday to Saturday) separately. If you select both options on the same panel, your scheduled job would not work even though the Scheduler accepts your setting.
12. The final **Add Scheduled Job** window is shown in Figure 45 on page 56.

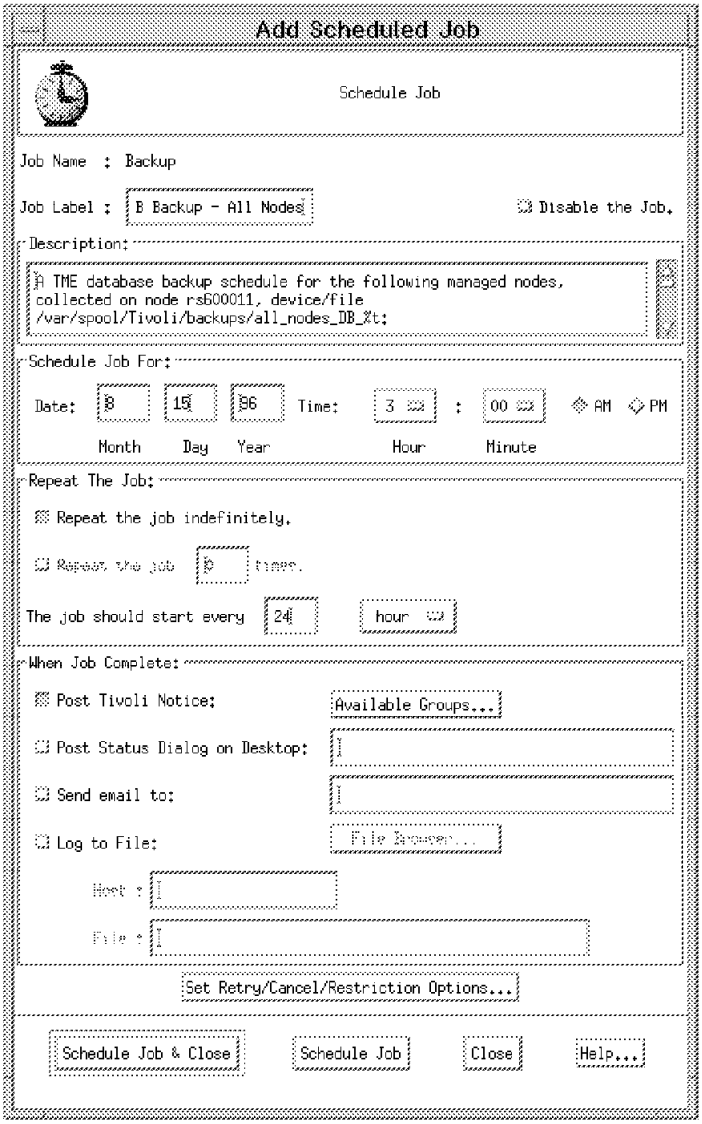


Figure 45. Add Scheduled Job Panel

13. Click on **Schedule Job & Close**.

Your scheduled backup can be seen in the Browse Scheduled Jobs panel of the scheduler (see Figure 46 on page 57).

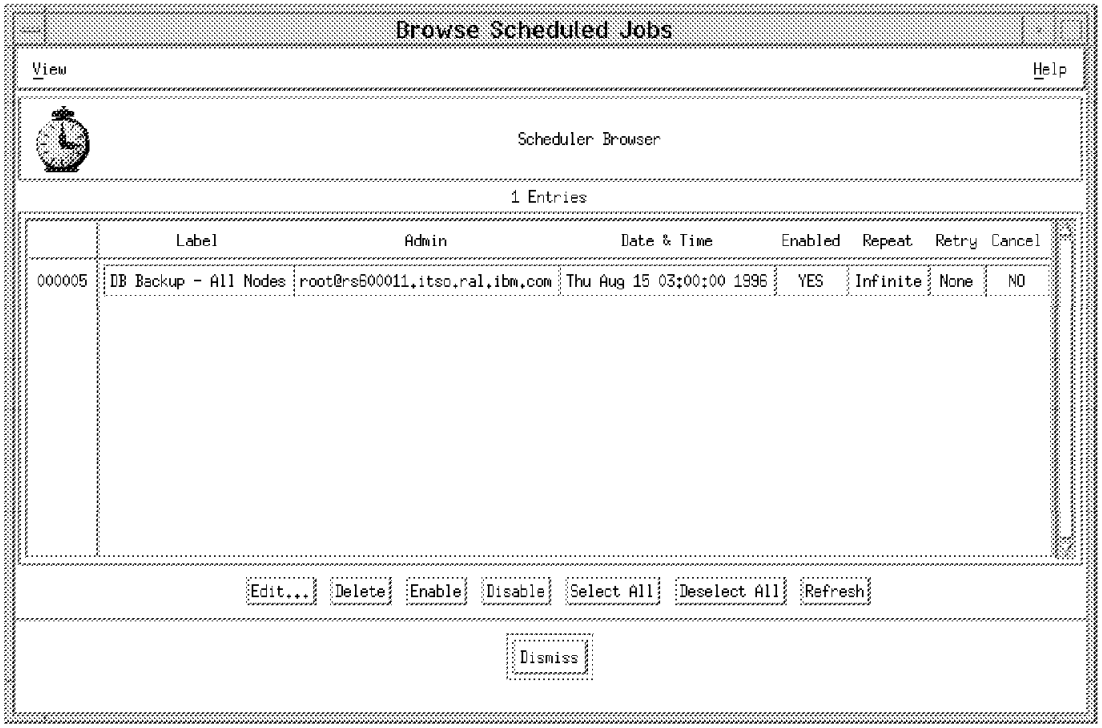


Figure 46. Browse Scheduled Jobs Panel

You can find the backup file under /var/spool/Tivoli/backups when your scheduled backup works correctly. You will receive a notice from the scheduler when this occurs.

Chapter 3. Configuring the TME Management Platform

To configure the Tivoli Management Environment in a way that best suits the management objectives of your organization, you need to have a management concept in place. This *TME Management Concept* must define at least the fundamental cornerstones of your Tivoli Management Environment, such as centralized or decentralized operation, sizes and boundaries of the Tivoli Management Regions and the management domains. Refer to 1.3, "A Practical Example of a TME Management Concept" on page 10 for an example.

The topics in this section guide you through the steps necessary to connect TMRs, to set up policy regions and to create TME administrators.

3.1.1 Connecting Tivoli Management Regions

To connect TMRs you need to know the following:

- Type of connection, one-way or two-way, secure or remote:
 - If it is a one-way connection, define which TME server is the *managing* server and which server is the *managed* server.
 - If you make a secure connection you need the region numbers of both TMRs. To get the region number of the local region enter `odadmin`.
- Encryption level and encryption password for inter-region connection

In our example, we used two-way interconnect between the TME servers rs600011 (TMR Prod) and rs60004 (TMR SD). The encryption level is Simple. We performed the Remote connection procedure.

Remote versus Secure Connections

When using the secure connection approach, the connection process is performed locally on each TMR (or on the TME server in each TMP). When using the remote TMR connection approach, the connection process is run remotely from one of the TMRs you are connecting to the other through either the trusted host facility or the remote login access.

Deciding whether to connect TMRs using the remote or the secure connection procedure is a question of network security. In other words, decide whether it is acceptable to send a password over the network or allow trusted host access during the connection process.

Once the connection is established, there is no difference between a connection made using the secure connection procedure and one using the remote connection procedure.

1. Log in as `root` or as any other TME administrator with the `super` role.
2. Bring up the Tivoli desktop by entering `tivoli`. Refer to 2.4.3, "How to Start the Tivoli Management Environment" on page 28 for more information.
3. From the administrator desktop select **Desktop**. Then select **TMR Connections => Connect**.
4. Fill in the The Connect to a Remote TMR dialog. Figure 47 on page 60 shows an example.

© Copyright IBM Corp. 1996

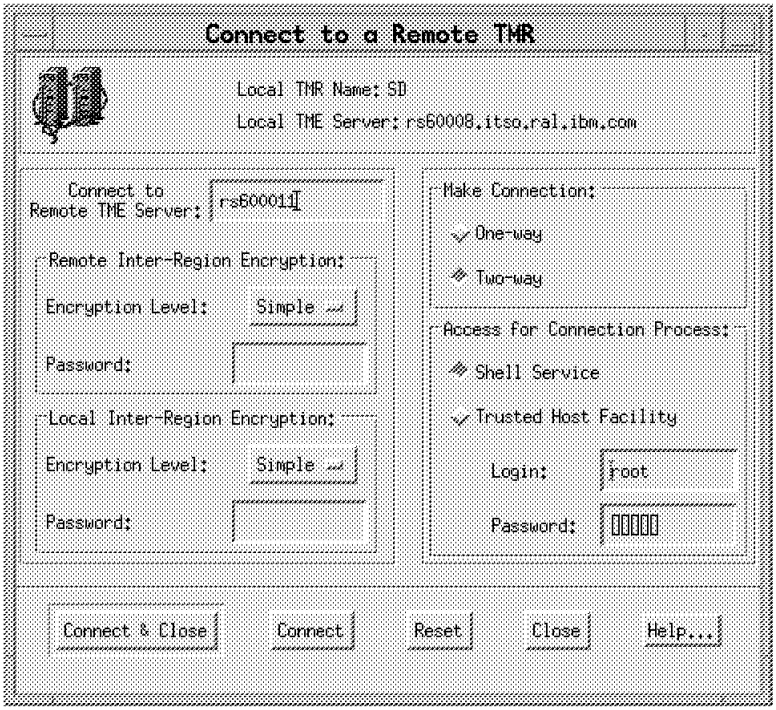


Figure 47. Connect to a Remote TMR

- Enter the host name of the remote server.
- Fill in the Remote Inter-Region Encryption and Local Inter-Region Encryption fields.

 Enter the same encryption level and the same password as specified during the TME server installation. Refer to Figure 8 on page 27 for an example of the Install Tivoli Server dialog.
- Select One-way or Two-way connection.

 In a one-way connection, only the managing node has information about the resources and roles in the other TMR, and management tasks can only be performed in one direction.

 In a two-way connection, both servers have information about the resources and roles in the other TMR, and management tasks can be performed in both directions.
- Choose one of the options in the Access for Connection Process field.

 Select **Shell Service** if you are going to provide a user ID and a password to access the remote server.

 Select **Trusted Host Facility** if this server can access the other TME server without providing a password. This means that this server is granted trusted host access on the remote server via an entry in the .rhosts file or via another authentication mechanism.

5. Select **Connect & Close** to start the connection process.
6. Choose whether you want to synchronize the *Name Registries* immediately. Figure 48 on page 61 shows an example of a Confirm Connect dialog.

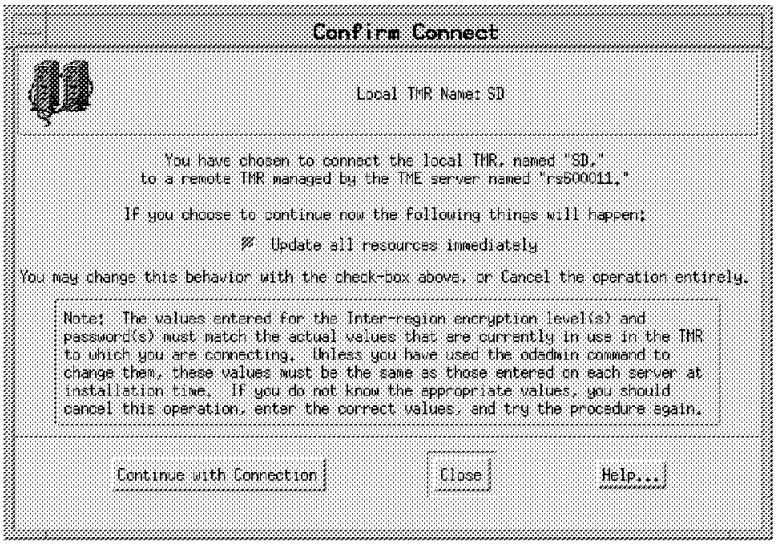


Figure 48. Confirm TMR Connection

If you would like the TME database synchronization to take place later, refer to chapter 3.1.4.1, “Synchronize TMR Databases” on page 65 for the procedure.

3.1.2 Create Policy Regions

There are two types of policy regions: *top level policy regions* and *subregions*. Top level policy regions are useful for organizing the resources to be managed into broad organizational categories, and are visible across TMRs.

Once the top level policy regions are defined, subregions under the top level policy region can be created to further subdivide administrative authority and policy.

The following example creates the top level policy region REGION1.

1. Log in as `root` or as any other TME administrator with the `senior` role.
2. Bring up the Tivoli desktop by entering `tivoli`. Refer to 2.4.3, “How to Start the Tivoli Management Environment” on page 28 for more information.
3. On the TME desktop select **Create = > Region** and fill in the name in the Create Policy Region dialog as shown in Figure 49 on page 62.
4. Select **Create & Close**.

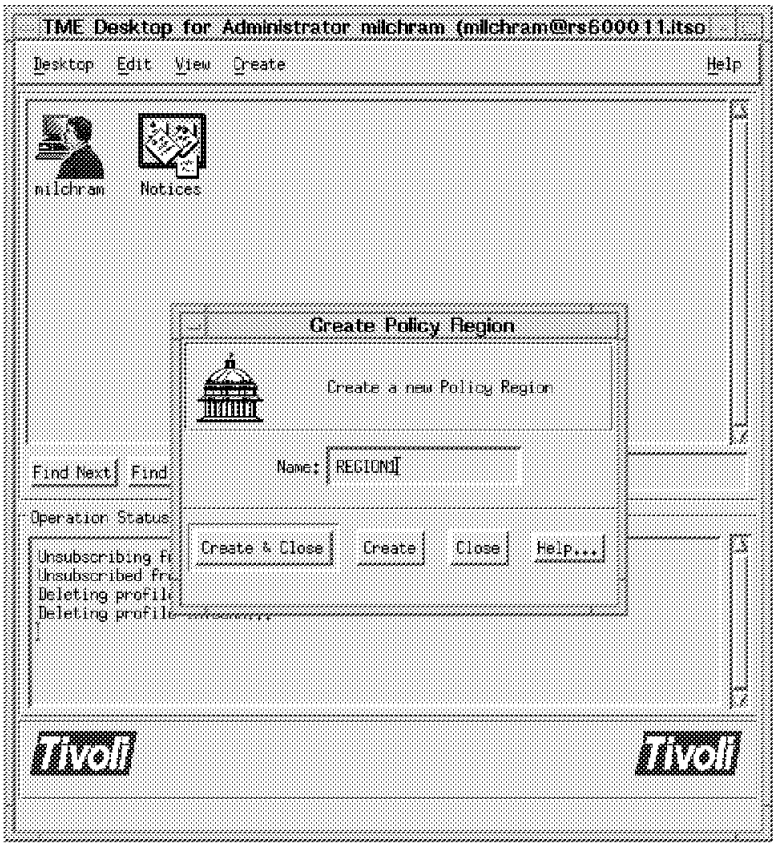


Figure 49. Create Policy Region

The only resources that a new policy region can handle are subregions. To enable a policy region to handle other resource types such as ManagedNode, InventoryProfile, etc., you have to add these resource types to the region's managed resources. This is done by double-clicking on the policy region icon. You then select **Properties => Managed Resources** and move the required resource types from the Available Resources area to the Current Resources area. Refer to Figure 10 on page 29 for an example of the Managed Resources dialog.

Subregions are created by double-clicking on the icon of the policy region that is going to contain the subregion. You then select **Create => Subregion** and fill in the Create Policy Region dialog as shown in Figure 49.

3.1.3 Arrange Policy Regions

A policy region is a collection of resources that share one or more common policies. Policy regions are abstract entities that are often used to model the organization of a distributed environment and are well suited to represent a management domain.

As shown in Figure 6 on page 19 we arranged our environment in such a way that the policy region Top was used as a container for policy regions Prod and SD. In other words, policy regions Prod and SD became subregions of the policy region Top.

This procedure shows how we arranged our top level policy regions to reflect the organization of our project.

1. Log in as root or as any other TME administrator with the senior role.
2. Bring up the Tivoli desktop by entering tivoli. Refer to 2.4.3, “How to Start the Tivoli Management Environment” on page 28 for more information.
3. From the administrator desktop select **Desktop**. Then select **TMR Connections => Top Level Policy Regions**. The Top Level Policy Region window as shown in Figure 50 is displayed.

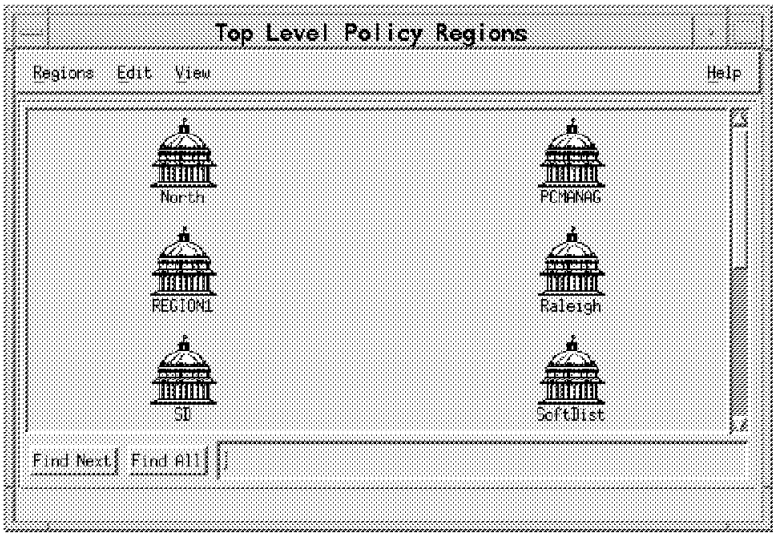


Figure 50. Top Level Policy Regions

4. Select the policy regions to be moved and drop them onto the policy region named Top.

 On the Top Level Policy Regions window select the policy regions (while holding down the Shift key, click and hold the left mouse button) and then drag and drop (release the left mouse button) them onto the policy region icon where they should be contained.

 If the operation was successful you will see a message similar to the following in the Tivoli desktop message area:

   ```
   The following resources were moved to the PolicyRegion named "Top".
           Prod (PolicyRegion)
           SD (PolicyRegion)
   ```

 Despite the above message indicating that the policy regions were moved, some or all of them might actually only get copied into the target collection. Remove them from the unwanted location on the desktop by selecting the policy region and **Edit => Remove**.

 Make sure that you don't accidentally delete them, because the Delete operation deletes a resource from the TME database whereas the Remove operation only removes it from the selected location on the TME desktop.

 Figure 51 on page 64 shows that the policy regions Prod and SD are now subregions of policy region Top.

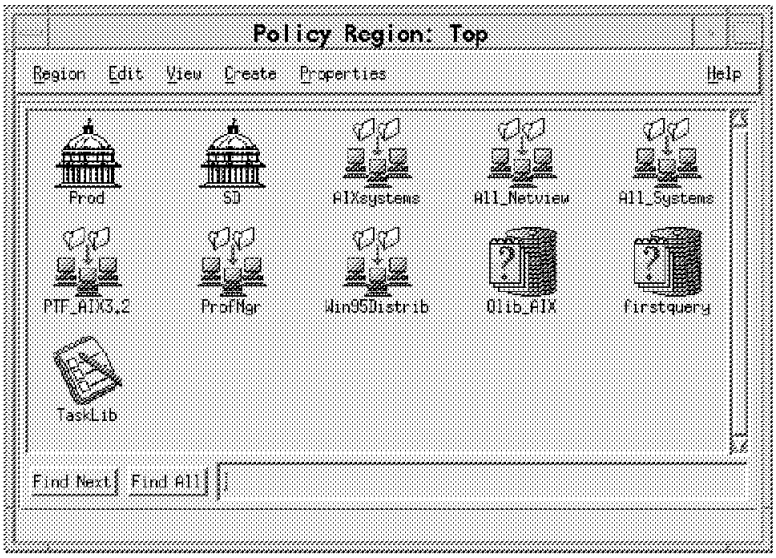


Figure 51. Subregions of Policy Region Top

How to Change a Subregion Back to Top Level

Top level policy regions can be arranged to become subregions of another policy region by dragging and dropping them onto the desired policy region icon. With our level of the Tivoli Management Platform we were unable to reverse this using the drag and drop functionality of the TME desktop. Instead, we had to use the command line.

The following example changes the subregion SD contained in the Top policy region back to a top level policy region by doing the following:

1. Linking it to the top level policy region collection.

 This means that the policy region icon SD exists now as a subregion of Top and in the top level policy region as well.

2. Removing the subregion SD from the desktop, as follows:

```
wln /Regions/Top/SD /Regions
:i1.wln
wrm /Regions/Top/SD
:i1.wrm
```

3.1.4 TMR Resource Updates

The *name registry* (or TMR database) is used as an intra-TMR name service and, when TMRs are connected, as an inter-TMR name service. To reduce the number of cross-TMR messages that must be sent during name lookups, the resource information of one TMR is maintained in the name registry of all connected TMRs.

During the initial connection process, the administrator is asked whether a resource update should be performed immediately upon connection. This is the only time that an update takes place automatically. To keep information on remote resources currently in the local name registry, updates must be scheduled at regular intervals.

TMR updates are always *pull* operations. This means that one TMR requests information from one or more connected TMRs, but that one TMR cannot *push* its current name registry resources to a connected TMR.

Frequency of Updates: After connecting TMRs, Tivoli recommends that you immediately exchange resource information between the TMRs. After this initial update, resource information should be updated on a regular basis. The frequency of these updates depends on the stability of your installation. For example, during the initial implementation of a Tivoli Management Environment, it might be necessary to run updates a few times per day. When the environment becomes more stabile, one update per day might be sufficient.

Updating Registries is Resource-Intensive

Updating registries is a very resource-intensive operation that could cause other performance problems. Therefore, Tivoli recommends resource updates not to be scheduled more frequently than once every one to two hours.

If a remote resource is needed immediately, initiate an update for this resource type only. Do not update all of the resource types.

3.1.4.1 Synchronize TMR Databases

Resource names are not updated automatically in the name registries of connected TMRs. Therefore, the names of remote resources can become stale. This means, for example, that a new resource created in a remote TMR is not registered in the local TMR′s registry, and thus cannot be used for local operations until the next update is performed.

A TMR database update is a pull operation, and must therefore be initiated from the TMR to be updated. This means that to update the registries of two-way connected TMRs, one update process must be started on each TMR.

One-way versus Two-way Connections

In a two-way connected environment, both TMRs have knowledge of the other TMR′s resources. In a one-way connected environment, only the *managing node′s* name registry is updated with the other TMR′s resources.

There are two ways to initiate a TMR database update:

1. Manual updates, using the TME desktop or the command line

 This method is mainly used to start updates if a remote resource is needed immediately.

2. Scheduled updates, using a scheduling function such as the TME Scheduler or UNIX crontab entries.

 Scheduled updates are used to keep name registries updated without administrator intervention.

Manual Updates: The following procedure uses the TME desktop and updates the TMR prod-region with information related to the top level policy regions of TMR SD:

1. Log in as `root` or as any other TME administrator with the `senior` role.

2. Bring up the Tivoli desktop by entering `tivoli`. Refer to 2.4.3, “How to Start the Tivoli Management Environment” on page 28 for more information.
3. On your TME desktop select **Desktop = > TMR Connections = > Update Resources**. This brings up the Update Resources from Multiple TMRs dialog as shown in Figure 52.

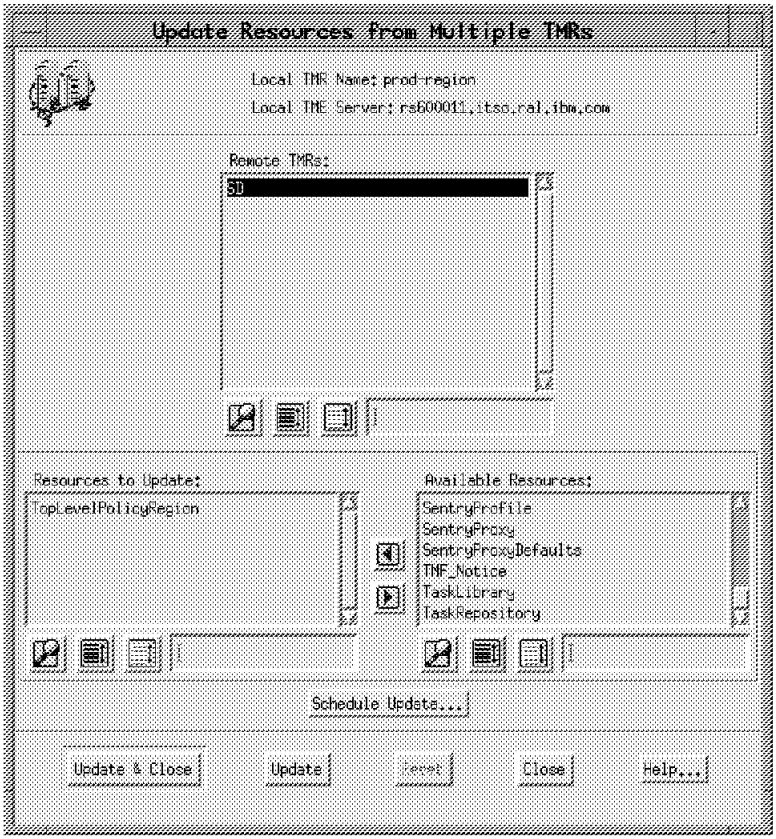


Figure 52. Update Resources from Multiple TMRs

- Select the remote TMR(s) to update from in the Remote TMRs area.
- Select the resources to update by moving them from the Available Resources area to the Resources to Update area.
- If you want to schedule the update by using the Tivoli Scheduler function, select the **Schedule Update** option. This brings up the Add Scheduled Job dialog as shown in Figure 53 on page 68. Refer to “Scheduled Updates” on page 67 for more details on how to use the Tivoli Scheduler facility to update TMR name registries.

4. Select **Update & Close**.

Command Line Examples: Updating name registries can also be done using the command line. Here are some update examples:

- This example updates the local TMR with information on the resource types TopLevelPolicyRegion and ManagedNode from TMR SD:

  ```
  wupdate -r TopLevelPolicyRegion -r ManagedNode SD
  ```

- This example updates the local TMR with information on all resource types from TMR SD:

  ```
  wupdate -r All SD
  ```

- This example updates the local TMR with information on all resource types from all TMRs:

  ```
  wupdate -r All All
  ```

Scheduled Updates: Updates can be scheduled either by using operating system-specific functions such as crontab entries in UNIX or by using the TME Scheduler facility. This procedure shows how to schedule a TMR name registry update that runs every day at midnight using the TME Scheduler.

Before you begin

Before you can use the TME Scheduler to run a job, there must be a task library, containing the task and the job to be executed.

Refer to 4.1.1, “Create Task Libraries” on page 81, 4.1.2, “Create Tasks” on page 82, and 4.1.3, “Create Jobs” on page 84 for information on how to create task libraries, tasks and jobs.

1. Log in as `root` or as any other TME administrator with the `admin` role.
2. Bring up the Tivoli desktop by entering `tivoli`. Refer to 2.4.3, “How to Start the Tivoli Management Environment” on page 28 for more information.
3. Open the policy region containing the task library where your update registry job can be found.
4. Drag and drop your job icon onto the TME Scheduler icon. The Add Scheduled Job dialog appears, as shown in Figure 53 on page 68. Fill in the required fields and select **Create & Close**.

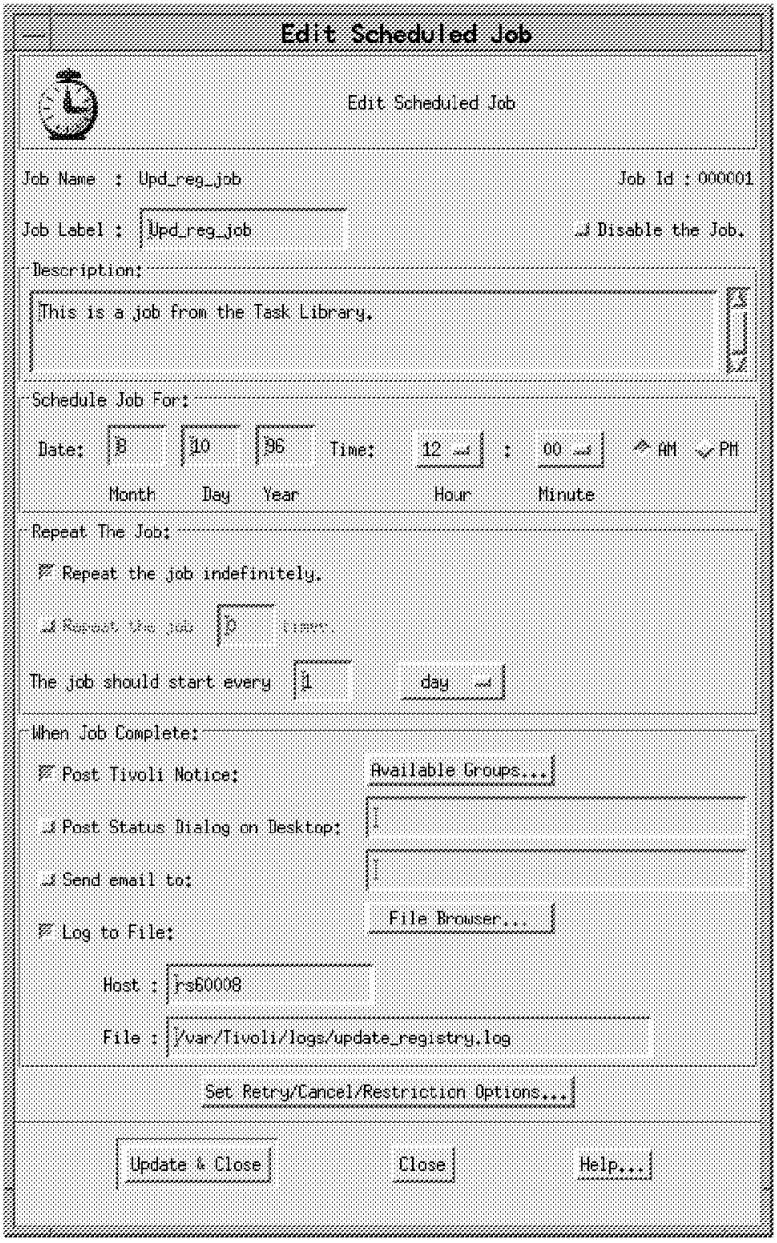


Figure 53. Add Scheduled Job

- Enter the date and time into the fields of the Schedule Job For area.
- Fill in the fields in the Repeat the Job area.
- Set the notice options in the When Job Complete area.

Note: For more information on the Scheduler function, refer to the *Tivoli Management Platform Planning and Installation User Guide*.

3.1.5 Create Profile Managers

Usually there are many profiles used to describe the entire configuration of a profile endpoint. Profile managers provide you with a convenient method of grouping profiles and other resources. Profile managers are not only used to group profiles and their subscribers; they also control the distribution of profiles to other profile managers or to profile endpoints across an entire network.

Defining a profile manager is a fairly simple task. The following example creates a profile manager called BigBoxes in policy region SD.

1. Log in as `root` or as any other TME administrator with the `senior` role.
2. Bring up the Tivoli desktop by entering `tivoli`. Refer to 2.4.3, “How to Start the Tivoli Management Environment” on page 28 for more information.
3. Select the policy region you want to contain the new profile manager by double-clicking on the policy region icon and select **Create => ProfileManager**.

 If there is no ProfileManager entry on the Create menu, you have to add ProfileManager as a managed resource type for the policy region. This is done by double-clicking on the policy region icon. Then select **Properties => Managed Resources** and move the ProfileManager entry from the Available Resources area to the Current Resources area.
4. Fill in the Name/Icon Label field in the Create Profile Manager dialog as shown in Figure 54.

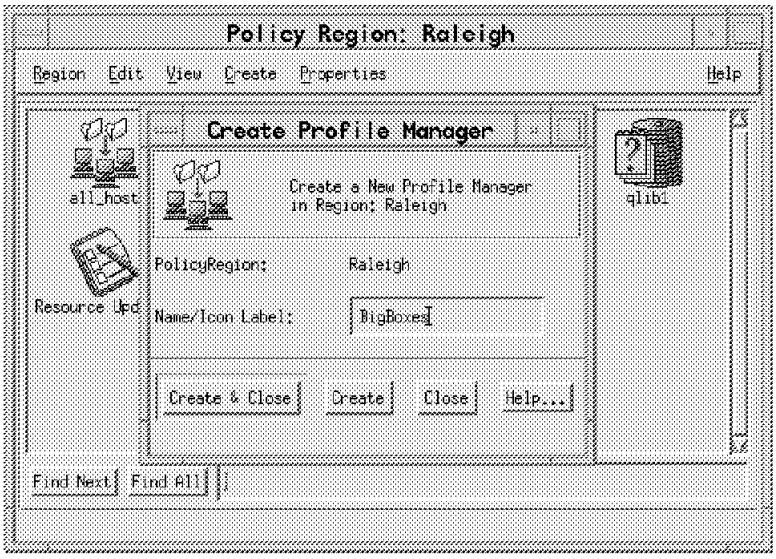


Figure 54. Create Profile Manager

3.1.6 Subscribe to Profile Managers

Subscribing profile managers or profile endpoints to a profile manager determines which resources will receive a profile when it is distributed.

Subscribers can be added in three different ways to a profile manager:

1. By *drag and drop*. With this method you select the subscribers and drop them onto the profile manager icon.
2. By *selection*. With this method you select the subscribers from a list of available subscribers.
3. By *command line*.

In the following example we use the selection method and add the profile endpoints rs60008 and rs60004 as subscribers to the profile manager BigBoxes.

1. Log in as `root` or as any other TME administrator with the `admin` role.

2. Bring up the Tivoli desktop by entering `tivoli`. Refer to 2.4.3, “How to Start the Tivoli Management Environment” on page 28 for more information.

3. Select the policy region containing the profile manger to which you want to add subscribers by double-clicking on the policy region icon.

4. Select the profile manager by double-clicking on the profile manager icon.

5. On the profile manager window select **Profile Manager = > Subscribers**. The Subscribers dialog shown in Figure 55 is displayed.

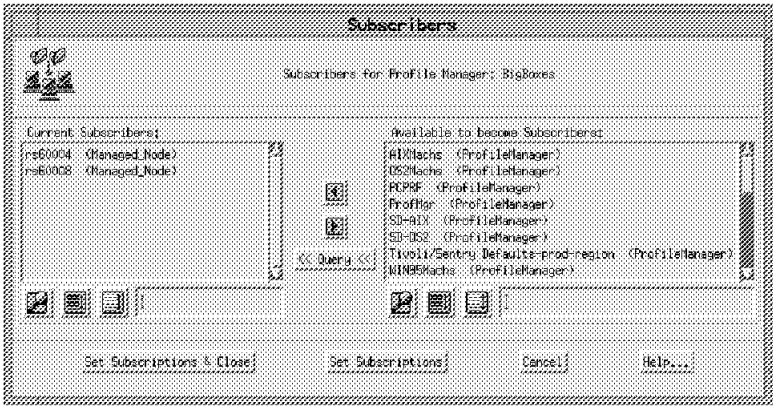


Figure 55. Add Subscribers to a Profile Manager

If you have Tivoli/Inventory installed you can select the **Query** button to run a predefined query against the available subscribers. For example, this could be used to select all systems from the available subscribers that run Windows 95. Refer to 6.5, “TME Query Facility” on page 235 for more information on the Query facility.

3.1.7 Create Administrators

The installation of the Tivoli Management Environment is performed by the user root. Therefore, root becomes the initial Tivoli administrator.

The default privileges of the root user comprises *super* authorization. The super role is only necessary to connect/disconnect TMRs and for other high security operations. The super role is not mapped to the super role in connected TMRs; it is mapped to the *user* role instead. Therefore, it is not necessary nor is it advisable to use root or any other administrator with the super role for day-to-day administration tasks.

Don´t use root for daily business

In a Tivoli Management Environment you can assign various authorization roles to any administrator, thus avoiding the need to use root for normal systems management tasks.

For our Tivoli environment at the ITSO, we applied the following administrator concept:

Root: The authorization roles of the root user were left unchanged. We used this administrator only when the super role was required.

TME Specialist: The highest role assigned to this generic administrator was the *senior* role. An administrator with the senior role can create and define all TME resources and is able to create other administrators.

We used this administrator to perform similar tasks in TME as one would use the root user in a traditional UNIX environment.

TME Administrator: The highest role assigned to this generic administrator was the *admin* role. An administrator with the admin role can perform day-to-day systems management tasks such as pushing a file package or adding a user item to a profile.

We used this administrator for day-to-day operation tasks.

Table 3. Roles on Resource Level

Administrator	**Resource Roles**		
	Administrators	**Scheduler**	**Region**
TME Specialist	n/a	n/a	n/a
TME Administrator	• admin • user • install_client	• admin • user • install_client	• admin • user • install_client

Table 4. Roles on TMR Level

Administrator	**TMR Roles**	**Notice Groups**	**Desktop**
TME Specialist	• senior • admin • user • install_client • install_product • backup • restore	all	• Administrators • Notices • Scheduler • Region
TME Administrator	• user • backup • install_product	all	• Administrators • Notices • Scheduler • Region

There are different approaches of providing an administrator with the desired roles in a TME:

- **Individual Definition**
 This means that every administrator is assigned roles on an individual basis. It has the advantage that every administrator will have a personal desktop, but the task to individually assign roles can be quite time consuming. If multiple TME administrators must be defined individually, we recommend that you automate this process by using a script.
- **Generic Definition**
 This means that administrators will be assigned roles in the TME by adding their login names to a generic TME administrator′s logins. It has the disadvantage that all administrators assigned to this generic TME administrator have the same desktop, but the definition process is quick. All defined roles are consistent and alterations to those roles can be done to the generic administrator.

To manage our TME environment, we used the generic definition approach.

3.1.7.1 Creating Administrators Using the Graphical Interface

The Tivoli Management Environment contains an easy-to-use graphical interface to perform most of the systems administration tasks necessary in a TME environment.

The following example shows how to create the generic TME administrator using the TME graphical user interface. Refer to 3.1.7, “Create Administrators” on page 70 for more information on the roles and the use of this administrator account.

1. Log in as `root` or as any other TME administrator with the `senior` role.
2. Bring up the Tivoli desktop by entering `tivoli`. Refer to 2.4.3, “How to Start the Tivoli Management Environment” on page 28 for more information.
3. Bring up the administrator window by double-clicking on the **Administrators** icon on the TME desktop.
4. Select **Create = > Create Administrator** to display the Create Administrator dialog as shown in Figure 56.

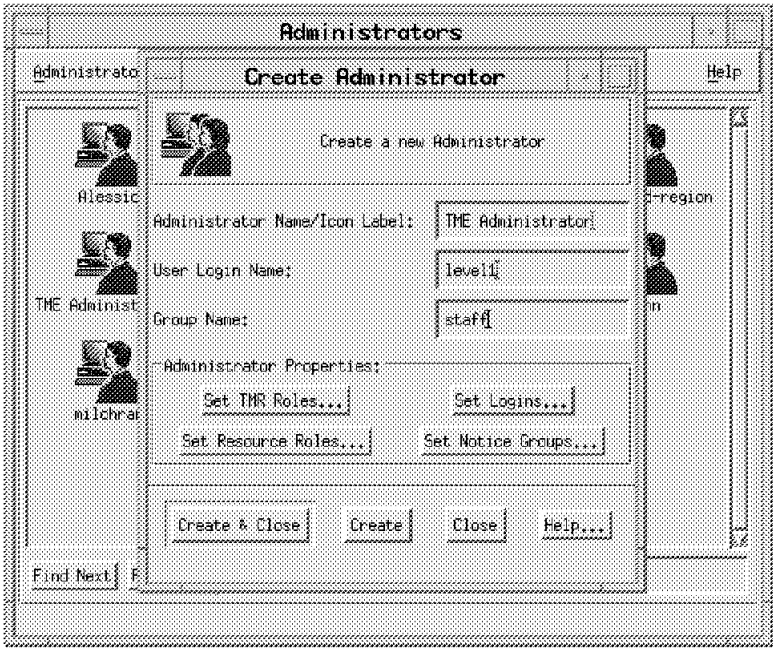


Figure 56. Create Administrator Dialog

- Enter the name of the administrator in the Administrator Name/Icon Label field.

 This is the name displayed with the administrator icon on the TME desktop.

- Enter the administrator’s user login name (not user ID) in the User Login Name field.

 This must be a valid user name on all machines managed with this TME administrator account, because various operations will be performed with the user ID derived from this user name. For example, an operation such as Run Xterm on a managed node will fail if this user name cannot be resolved to a valid user ID on the managed node.

- Enter the administrator's group name (not group ID) in the Group Name field.

 This must be a valid group name on all machines managed with this TME administrator account, because the group ID derived from this group name will for example be used for all output operations to a file.

5. Select **Set TMR Roles** and select the roles for this new administrator as shown in Figure 57.

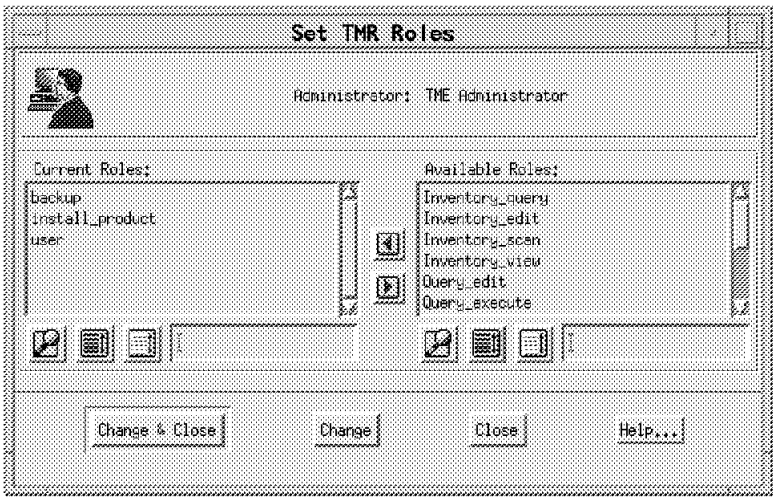


Figure 57. Set TMR Roles

Selecting roles is done by moving them (double-clicking) from the Available Roles area to the Current Roles area on the Set TMR Roles window.

6. Roles given to an administrator on TMR level are applicable to all resources in this TMR. TMR roles are only required for TMR-wide operations. Therefore, roles should be assigned on an individual resource level.

7. Select **Set Resource Roles** and select the roles for this new administrator as shown in Figure 58 on page 74.

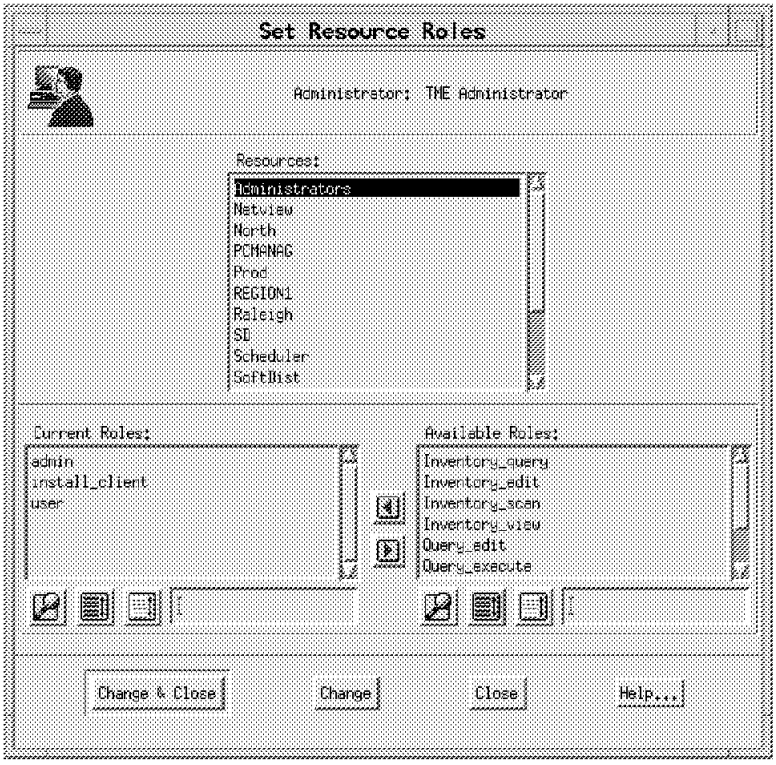


Figure 58. Set Resource Roles

If you want, for example, to enable an administrator to create other Tivoli administrators or to schedule operations, the administrator must have one or more roles over the Administrators collection or the Scheduler resource.

8. Select **Set Logins** to bring up the Set Login Names dialog as shown in Figure 59.

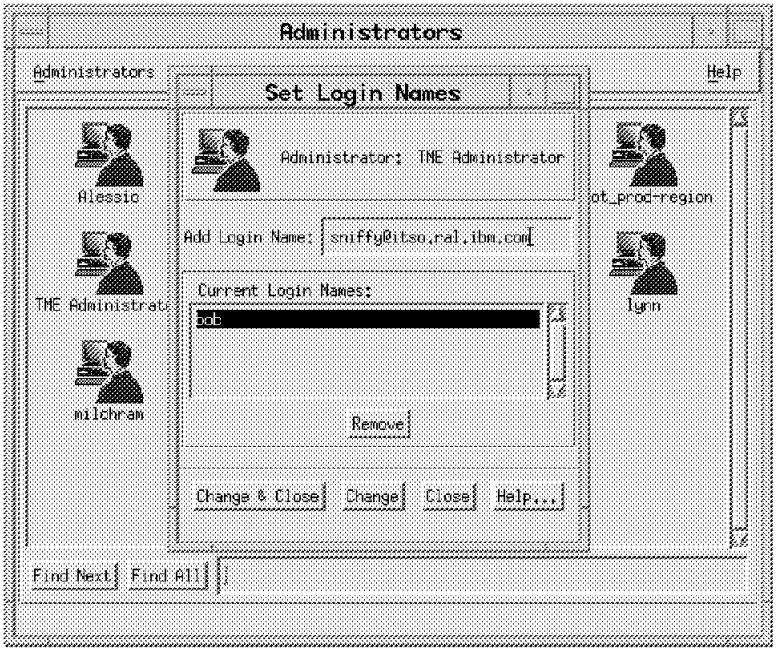


Figure 59. Set Login Names

Enter the login names in the Add Login Names field, under which you want the new administrator to be able to start the TME desktop:

- A user name specfied as user_name will allow this user to start the Tivoli desktop from any system within the local TMR.
- A user name specified as user_name@hostname will allow this user to start the Tivoli desktop only from one particular system within the local TMR.

If you use generic administrators, this is the place to add all of the login names that should have the same administration roles.

9. Select **Set Notice Groups** to bring up the Set Notice Groups dialog as shown in Figure 60 and subscribe the administrator to the desired notice groups.

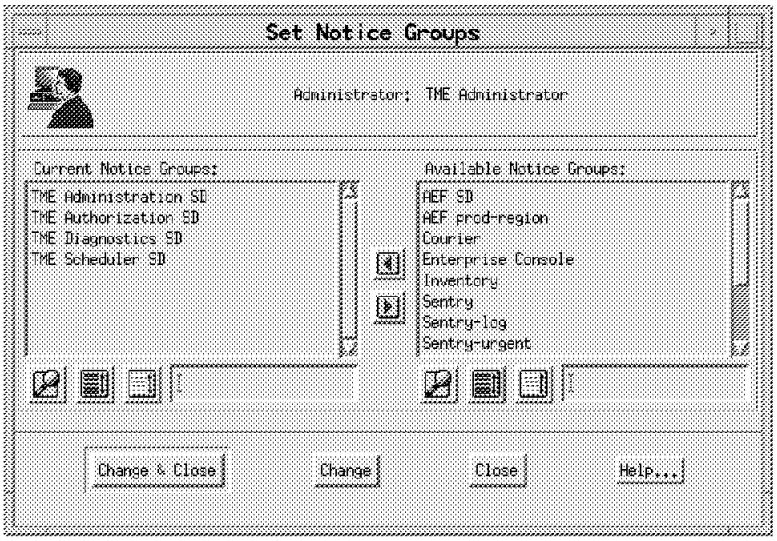


Figure 60. Set Notice Groups

Notices are generated when TME management functions are performed and are sent to an operation-specific *notice group*. This means that an administrator, for example, will only receive messages related to the creation of a new administrator if the administrator is subscribed to the TME Administration notice group.

10. Click on **Select & Close** to create the new administrator.
11. Add resources to the administrator's desktop by dragging and dropping the required resources onto the new administrators desktop.

 When new administrators are created, they will only have the Notices icon on their desktop. To assign them a set of resources to manage from their TME desktop, the icons representing these resources must be copied to the new administrator's desktop:

 - Open the Administrators collection by double-clicking on the **Administrators** collection icon.
 - Select the resources to copy and drag/drop them onto the new administrator's icon in the Administrators collection.

3.1.7.2 Creating Administrators Using the Command Line Interface

The easiest way to create multiple individual administrators with the same or similar set of roles is to create them by means of a script. Figure 61 shows how to create a Tivoli administrator with wcrtadmin from the command line or with a script.

```
wcrtadmin -l test_adm \
  -l sniffy@rs60008.itso.ral.ibm.com \
  -n "TME Administration" \
  -n "TME Authorization" \
  -r global,user:admin:backup \
  -r /Administrators,user:admin:senior \
  -r /Scheduler,user:admin:senior \
  -r @PolicyRegion:sd-region,user:admin:senior \
  -u test_adm -g staff "Muhammad Ali"
```

Figure 61. Create Tivoli Administrator from Command Line

The example in Figure 61 creates an administrator with the following roles and attributes:

Login Names	test_adm sniffy@rs60008.itso.ral.ibm.com	
Notice Groups	TME Administration TME Authorization	
TMR Roles	admin user backup	
Resource Roles	Administrators	senior admin user
	Scheduler	senior admin user
	Region	senior admin user
Principal user name	test_adm	
Principal group name	staff	
Administrator label	Muhammad Ali	

3.2 Some Performance Considerations within the TME Environment

This section covers the performance analysis for the TME region used for the development of this redbook. The performance figures shown here represent the ITSO TMR configuration and should not be used for sizing purposes.

The performance data gathering process used here could be adapted to suit any particular RS/6000 environment.

We set up a the Performance toolkit to monitor the memory requirements for the TME 10 environment. The data was collected from a number of the servers within

the ITSO TME 10 framework. We wanted to see what the impact was to the RS/6000 resources before, during and after starting the TME applications.

The hard disk requirements for each of the machines can be found in the installation chapter in this book.

The TME application servers monitored are listed below:

rs600024 The TMR Server

rs600021 A Managed Node

rs600020 The T/EC Server

3.2.1 Installation of the Performance Toolkit

The Performance toolkit server was loaded on rs600021. All of the other RS/6000 machines have the agent loaded and configured.

One performance toolkit console was created for the purpose of data collection, and is named:

- TME_MEMORY_STATS

3.2.2 Approach

The performance data was collected for a number of different phases for each machine. This enabled us to calculate the impact of each software component on each system. The data collected was for the overall system requirements. We did not monitor any particular processes.

The memory configuration for each processor is listed below:

- rs600024
 - Paging space 200 MB
 - RAM 128 MB
 - Hard disk space 2.2 GB
- rs600020
 - paging space 256 MB
 - RAM 128 MB
 - hard disk 4.5 GB
- rs600021
 - Paging space 280 MB
 - RAM 128 MB
 - Hard disk 2.2 GB

In Figure 62 on page 78 you can see the performance variables that we monitored. These are detailed below:

Real/noncomp Non-computational memory shows the shared program memory.

Real/comp Computational memory shows the actual private memory used.

PageSp/totalfree The actual free paging space.

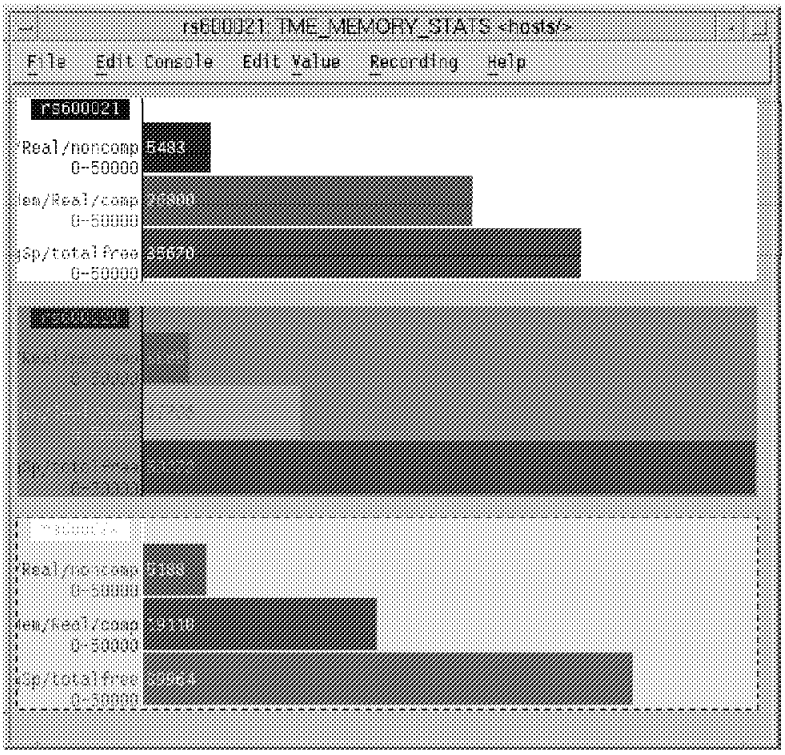


Figure 62. Performance Data Capture Values

The data was collected for a number of phases. These phases represent different stages of starting up the TME 10 applications:

Start Recording data Record data in file /u/paul/XmRec/R.TME_MEMORY_STATS.

phase 1 Base system Monitor all three systems to get the baseline values.

Phase 2 on rs600024 Run odadmin start (start the Tivoli management server).

Phase 3 on rs600021 Run odadmin start (start the Tivoli daemons).

Phase 4 on rs600021 Run Tivoli (start the Tivoli desktop).

Phase 5 on rs600020 Run odadmin start (start the Tivoli Daemons & T/EC Server).

Phase 6 on rs600021 Start the T/EC console paul from the Tivoli desktop.

Phase 7 on rs600020 Run Tivoli (start the Tivoli desktop).

Phase 8 on rs600020 Start the T/EC console root from the Tivoli desktop.

Phase 9 on rs600024 Run Tivoli (start the Tivoli desktop).

End of phases Stop recording data.

During the data capture, we allowed a five-minute gap to let the systems settle down after the initial hit on memory.

The capture file in our case was called:

```
/u/paul/XmRec/R.TME_MEMORY_STATS
```

The capture file was then analyzed using the following commands:

```
ptxtab R.TME_MEMORY_STATS
```

This command creates one ASCII file for each of the three monitored machines. In our case, the files are named:

```
A.TME_MEMORY_STATS_02
A.TME_MEMORY_STATS_03
A.TME_MEMORY_STATS_04
```

The data captured was then analyzed.

3.2.3 Analyzing the Data

Once the data was converted into ASCII format we evaluated the output against the phases. Each phase was time stamped.

The data captured represents values of 4 KB blocks of memory. For example, the line below shows an extract from the data collected for rs600020.

```
Monitor: TME_MEMORY_STATS --- hostname: rs600021
                            Mem       Mem
                           Real      Real     PagSp
Timestamp               noncomp      comp totalfree

1996/09/10 11:51:01        8742     22932     36126
1996/09/10 11:52:00        8864     23774     35103
```

Figure 63. Snapshot of Data Collected

The evaluation process was to calculate the overall memory required for each phase. For example, the first line in Figure 63 shows the system rs600021 without the Tivoli daemons running, and the second line with the Tivoli daemons running.

Here we can observe an increase in both computational and non-computational memory, while there was a decrease in the available paging space.

In each of the nine stages, we allowed data to be captured at 20-second intervals for a total of five minutes. An average of the memory was taken for that period.

The findings from the data captured are detailed below.

The base figures show the following memory requirements for the systems, rs600024, rs600020, and rs600021, with no Tivoli applications running.

The impact on rs600024 when the odadmin start was executed is on paging space and overall RAM.

Table 5 (Page 1 of 2). Performance Data Collection Phases

Phase	Machine	Description.
1	rs600024	odadmin start
2	rs600021	odadmin start
3	rs600020	odadmin start
4	rs600021	tivoli
5	rs600020	tivoli
6	rs600024	tivoli

Table 5 (Page 2 of 2). Performance Data Collection Phases		
Phase	**Machine**	**Description.**
7	rs600021	T/EC
8	rs600020	T/EC
9	rs600024	T/EC

3.2.4 Results of Data Analysis

Table 6. Data Analysis			
Phase	**Machine**	**RAM Required**	**Paging Space Used**
1	rs600024	2452	2008
2	rs600021	1244	1068
3	rs600024	14124	13156
4	rs600021	1996	2724
5	rs600020	2228	3020
6	rs600024	2728	1720
7	rs600021	540	negligible
7	rs600020	2700	2414
8	rs600020	2880	1616
9	rs600024	902	negligible
9	rs600020	2412	1260

3.2.5 Observations

The following observations were derived from the captured data. The overall memory requirements for our environment are shown in Table 7.

Table 7. Overall Memory Requirements			
Node	**Shared Memory**	**Private Memory**	**Paging Space**
rs600020	negligible	22 MB	24 MB
rs600021	negligible	3.7 MB	6 MB
rs600024	negligible	12 MB	9.5 MB

The memory requirements for the T/EC server increase depending on the number of alerts generated. In our case, we have around 2000 events in the T/EC database.

Chapter 4. Task Libraries, Tasks and Jobs

The TME platform provides a number of built-in services, all of which are based around the object request broker architecture. However, in general you cannot actually *do* anything with the platform until you have installed an application that makes use of the base services. The notable exceptions to this rule are *task libraries*, *tasks* and *jobs*.

A task library is a feature of the TME that enables you to create and store tasks and jobs. Tasks and jobs are programs or commands that can be run on multiple machines in the network as required. Task libraries are defined in the context of a policy region and you can create multiple task libraries in different policy regions. This is useful if you want to create sets of tasks that are specific for a particular function, set of resources or group of administrators.

A task defines certain operations that usually have to be performed on a regular basis, such as clearing a printer queue, starting backups, etc. Each of these operations is routinely performed on various machines and even on various platforms on the network. A task defines the executable to be run, the TME role required to execute the task, etc. A task does not define the detailed information required to run the task, such as the systems to run on, what output type you would like, etc. This information must be provided when the task is executed.

A job is a task that is executed on a specifically managed resource. When you create a job you select an already defined task to be executed and you define the execution information required to execute the task. Once a job has been created you can run the job without providing any further information.

4.1 A Simple Task Library Example

First we show a simple example of creating a task and a job in a task library. Later we look further at the requirements needed to set up administrators for task execution.

4.1.1 Create Task Libraries

In this example we created a task library called Resource Updates in the policy region Raleigh:

1. Log in as `root` or as any other TME administrator with the `senior` role.
2. Bring up the Tivoli desktop by entering `tivoli`. Refer to 2.4.3, “How to Start the Tivoli Management Environment” on page 28 for more information.
3. Select the policy region you want to contain the task library by double-clicking on the policy region icon and select **Create => TaskLibrary**.

 If there is no TaskLibrary entry on the Create menu, you have to add TaskLibrary as a managed resource type for the policy region. This is done by double-clicking on the policy region icon. Then select **Properties => Managed Resources** and move the TaskLibrary entry from the Available Resources area to the Current Resources area.
4. Fill in the **Name/Icon Label** field in the Create Task Library dialog and select **Create & Close**. This creates the new task library within the selected policy region as shown in Figure 64 on page 82.

© Copyright IBM Corp. 1996

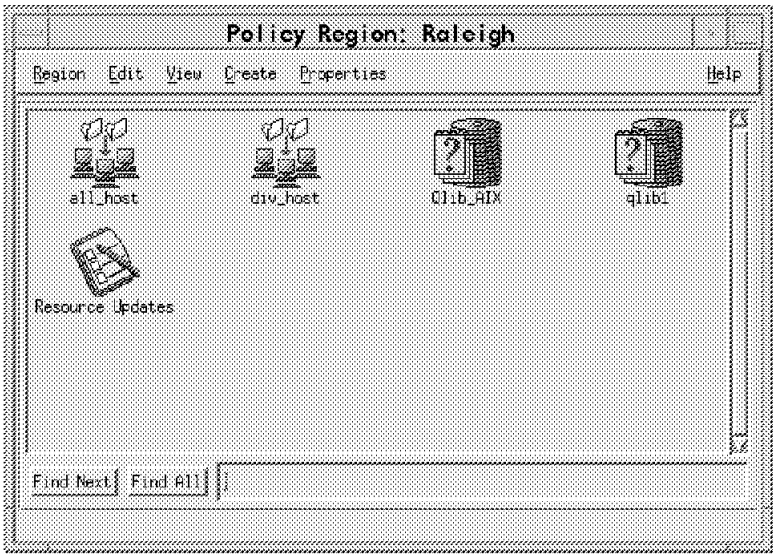


Figure 64. Task Library

4.1.2 Create Tasks

In this example we created the task Upd_reg_tsk within the task library Resource Updates:

1. Log in as `root` or as any other TME administrator with the `admin` role.
2. Bring up the Tivoli desktop by entering `tivoli`. Refer to 2.4.3, "How to Start the Tivoli Management Environment" on page 28 for more information.
3. Open the policy region containing the task library in which you want to create the task. Then double-click on the task library icon.
4. On the Task Library window select **Create = > Task**. The Create Task dialog as shown in Figure 65 on page 83 appears. Fill in the required fields and select **Create & Close**.

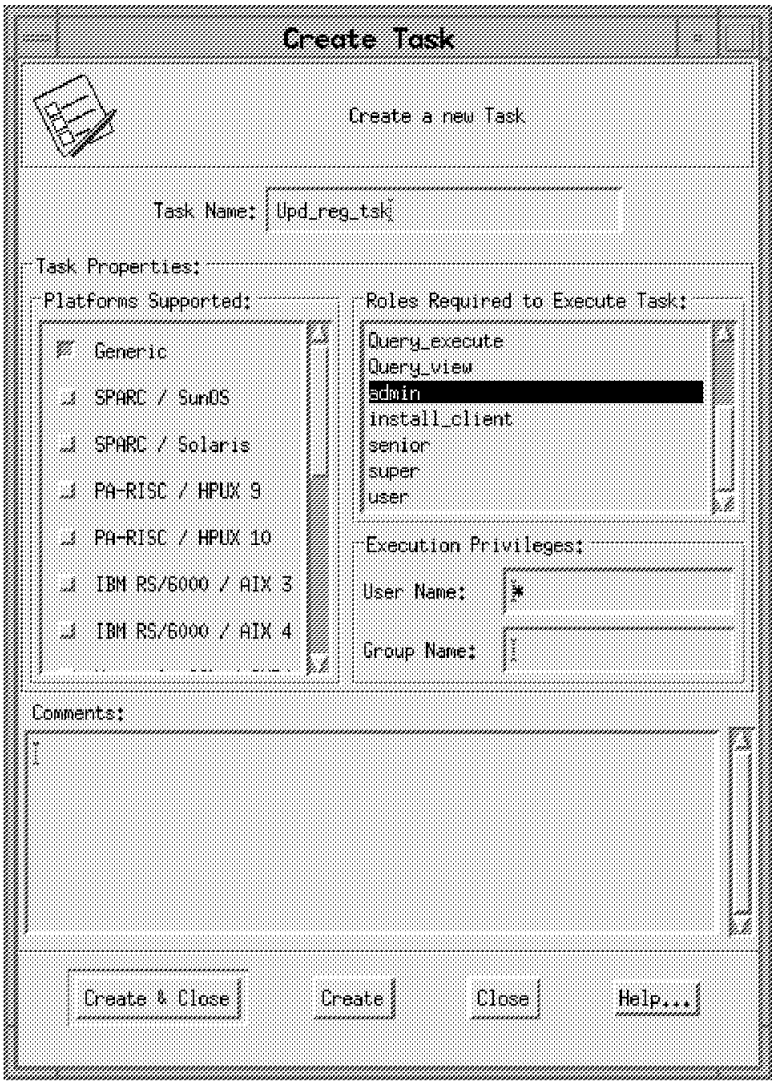


Figure 65. Create Task

- Select the platform for which you want to create the task in the Platforms Supported area.

 Multiple platforms can be selected. You will have to provide the host name and the path to the executable for each platform you select.

 If you have a generic executable, such as a shell script, that can run on multiple architectures, select **Generic**. Otherwise, select the matching platform / operating system combination.

- Select the roles required to run the task from the Roles Required to Execute Task area.

- If you want the task to run under a specific user ID or group ID fill in the fields in the Execution Privileges field. If you leave these fields unchanged the task will run under the ID of the administrator executing the task.

4.1.3 Create Jobs

In this example we created the job **Upd_reg_job** within the task library **Resource Updates**:

1. Log in as `root` or as any other TME administrator with the `admin` role.
2. Bring up the Tivoli desktop by entering `tivoli`. Refer to 2.4.3, "How to Start the Tivoli Management Environment" on page 28 for more information.
3. Open the policy region containing the task library in which you want to create the job. Then double-click on the task library icon.
4. On the Task Library window select **Create = > Job**. The Create Job dialog as shown in Figure 66 appears. Fill in the required fields and select **Create & Close**.

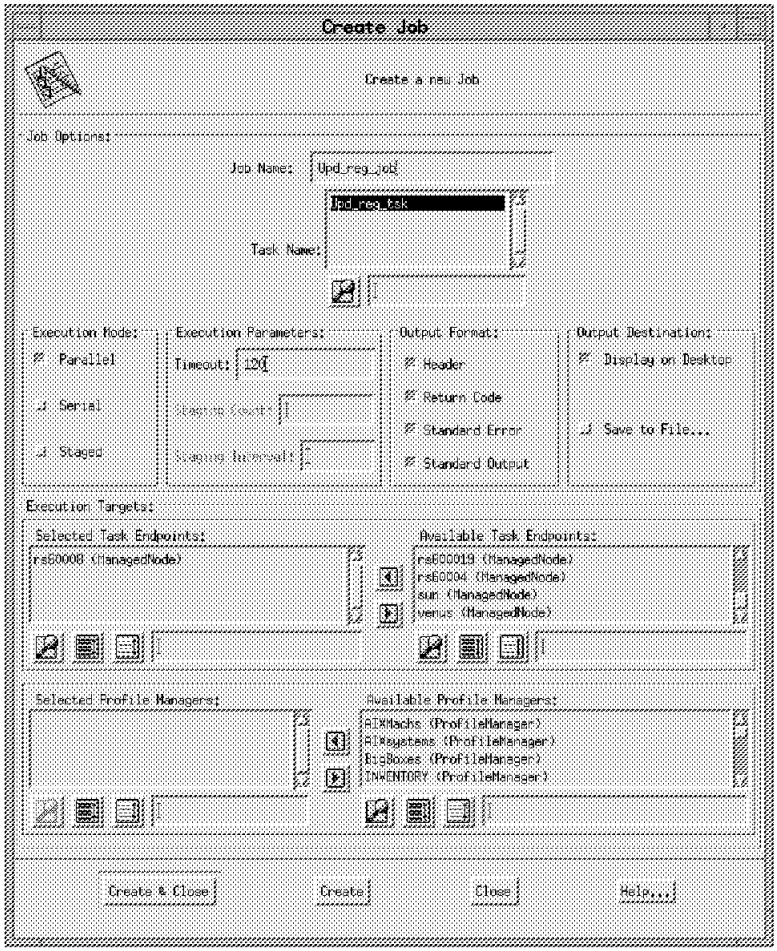


Figure 66. Create Job

- Select the required Execution Mode.

 Parallel means that this job will run concurrently on all systems.

 Serial means that this job will run sequentially on all systems.

 Staged means that this job will run in staged sets. If you select staged, also select the staging count and the staging interval.

- Enter the timeout value (seconds) for the job. If the job does not complete in the given period of time, any output from this job will be lost.

- Select the Execution Targets by moving them from the Available area to the Selected area.

 You can select systems from the Available Task Endpoints area or profile managers from the Available Profile Manager area or a combination of both. If you select a profile manager, all systems subscribing to this profile manager will be execution targets for this job.

4.2 A More Complex Task Example

In this example, we look in more detail at the task facility, with the aim of tying together the steps needed to use them productively. Some of the areas we cover include:

- The authorization roles that are required by administrators to execute tasks
- How to execute tasks from the command line
- How to create tasks that require user input

4.2.1 Create an Administrator

First we define a new TME administrator ID to execute the tasks that we create.

Administrator lynn will be created with the authorization roles required to create tasks:

1. From root′s Desktop, select **Administrators = > Create Administrator**.
2. Enter the administrator name, the login name and the group name.

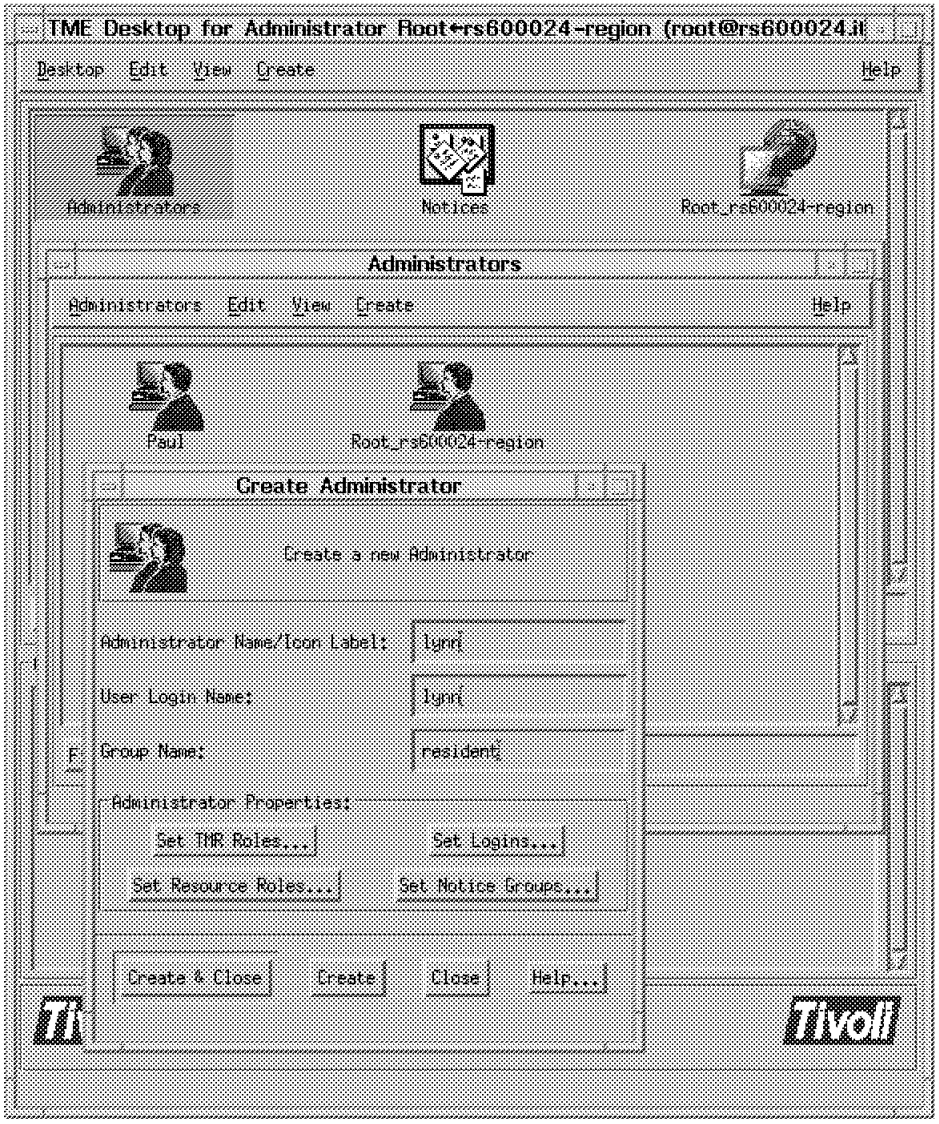


Figure 67. Create a New Administrator

3. Select **Set TMR Roles**. If lynn's desktop will be run on a client, the minimum authority required to start the desktop is user. If the desktop is going to be run only on the server, a TMR role is not required.

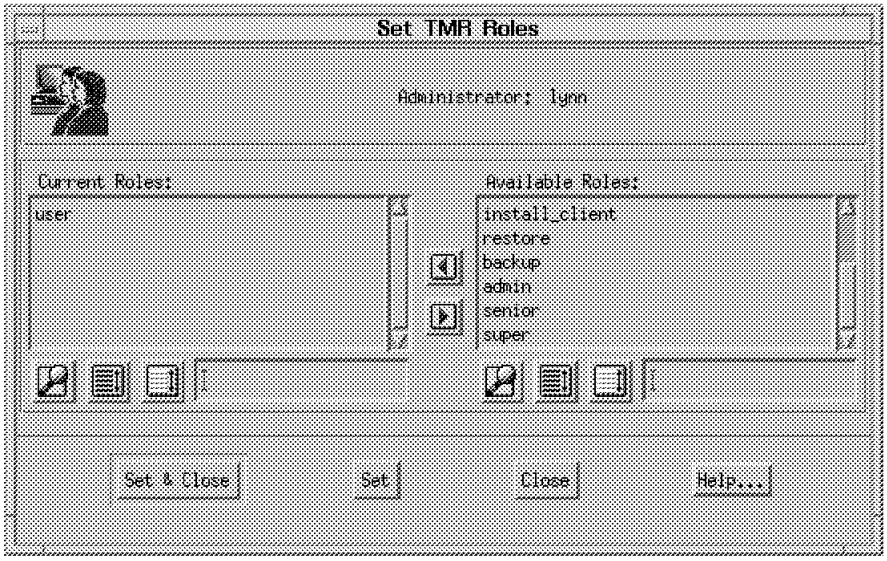


Figure 68. Set TMR Roles

4. Select **Set Resource Roles**. The authorization required to *create* tasks in a Task Library is admin.

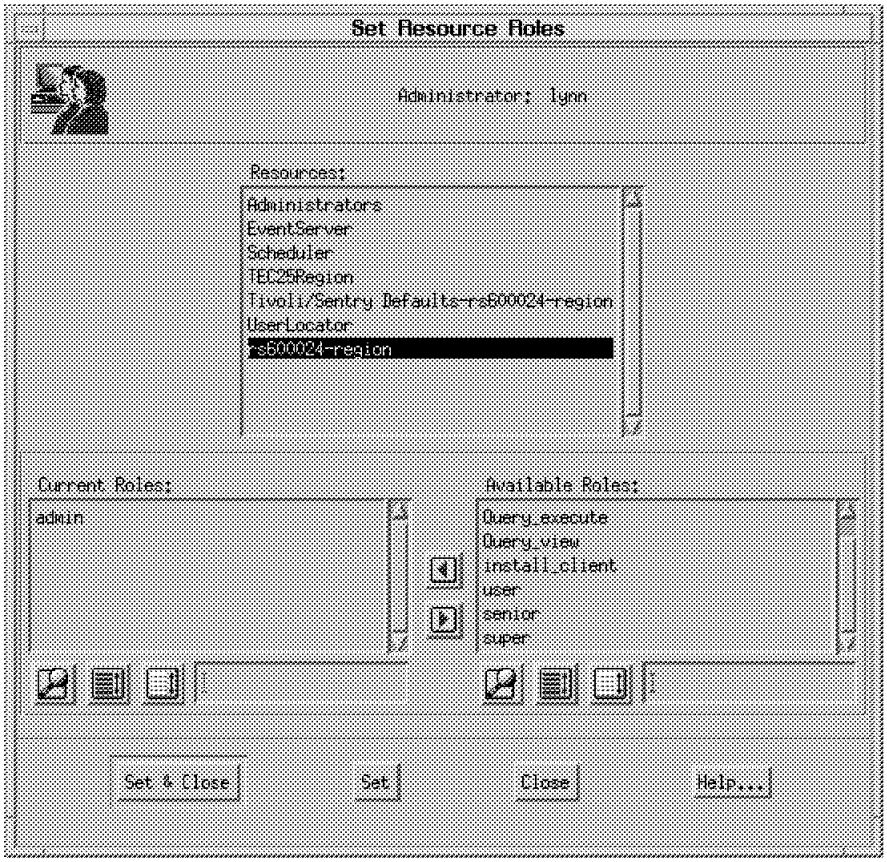


Figure 69. Set Resource Roles

5. Select **Set Logins**.
6. Enter the login name and optionally the name of the machine where this Administrator is allowed to start the desktop or run commands. Don't forget

to press Enter after keying in the name. If you click on **Set & Close** without first pressing Enter, the name is not saved.



Figure 70. Set Logins

7. At this stage, you do not need to select **Set Notice Groups**, so simply select **Create & Close**. Administrator lynn will appear on the Administrators window, as shown in Figure 71.

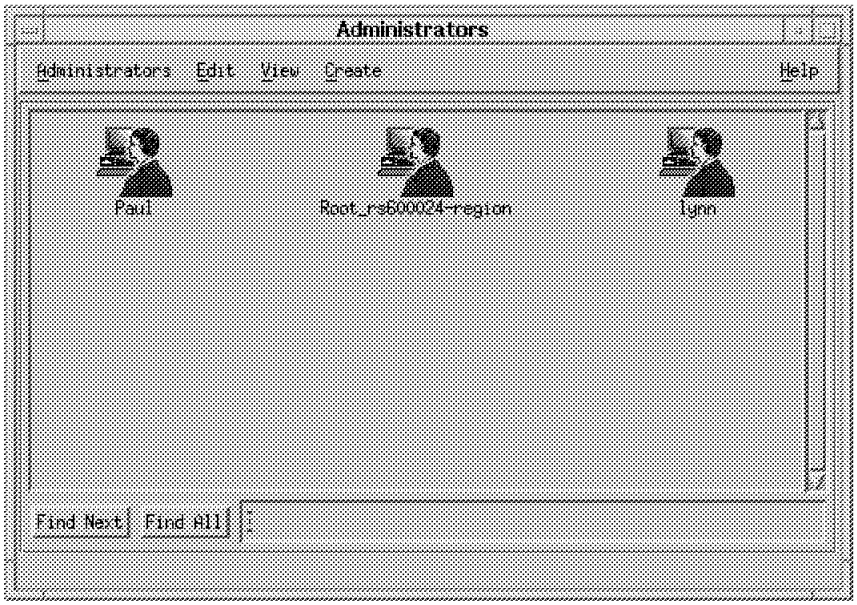


Figure 71. New Administrator Icon Appears

8. With the right mouse button, select the **lynn** icon, then select **Open** from the menu list or double-click on it with the left mouse button. This will open lynn's desktop.

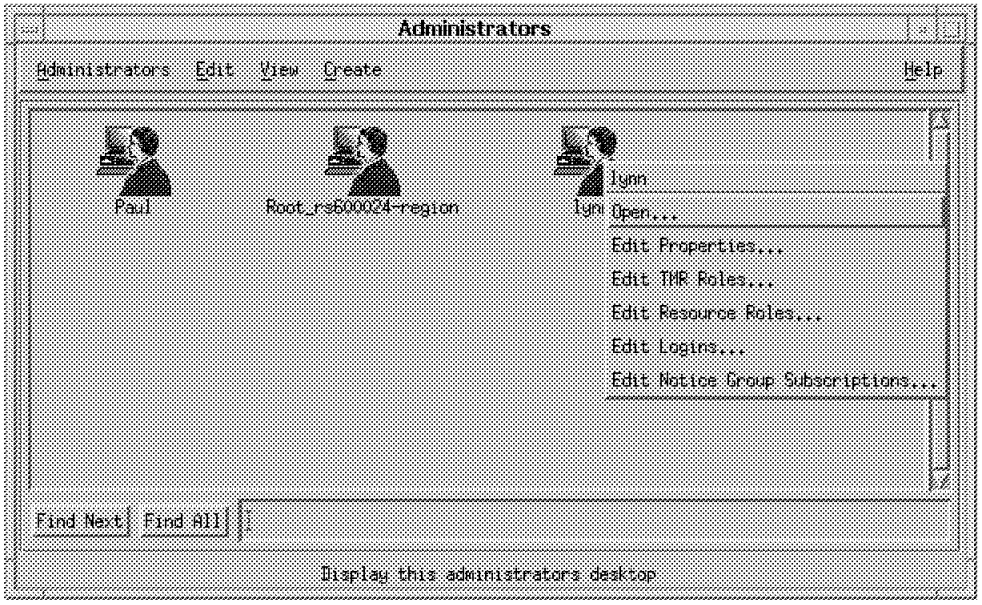


Figure 72. Open Administrator Desktop

9. With the left mouse button, drag and drop the Policy Region in which you want to create a task library (rs600024-region in our case) from root's desktop onto lynn's desktop.

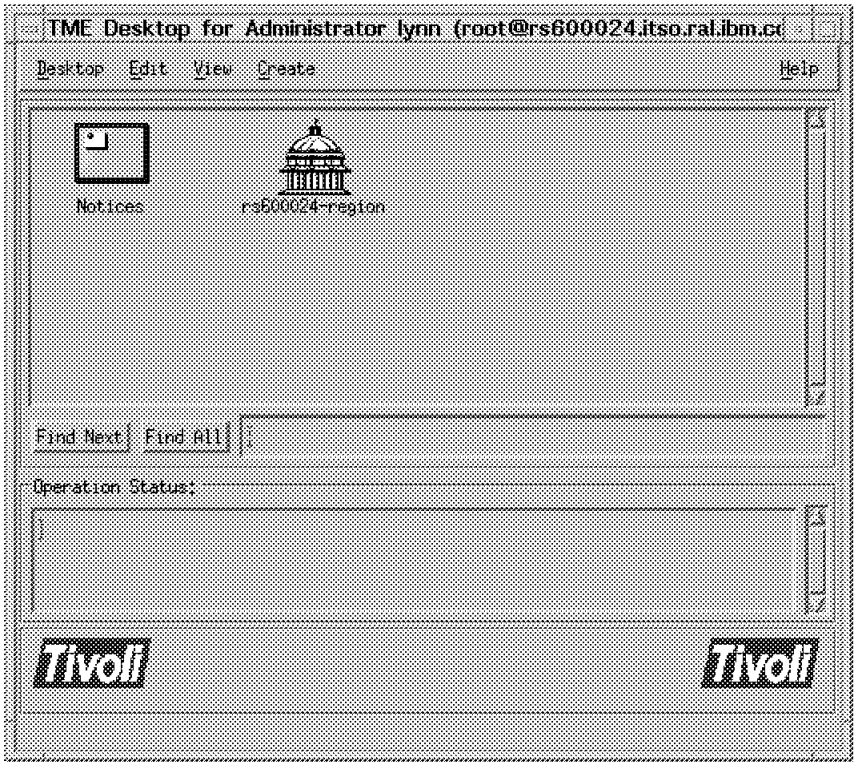


Figure 73. Lynn's Desktop with One Policy Region

10. On lynn's desktop, select **Desktop => Close**.

4.2.2 Create a Task Library

Creating task libraries requires *senior* authority in the Policy Region and at least *user* authority in the TMR. If the Administrator has no authority at the TMR level, the option **Task Library** will not appear under the **Create** menu. Administrator lynn has been given admin authority (to allow creation of tasks) at the Policy Region level and user authority at the TMR level. If lynn tries to create a task library, the message shown in Figure 74 will appear.

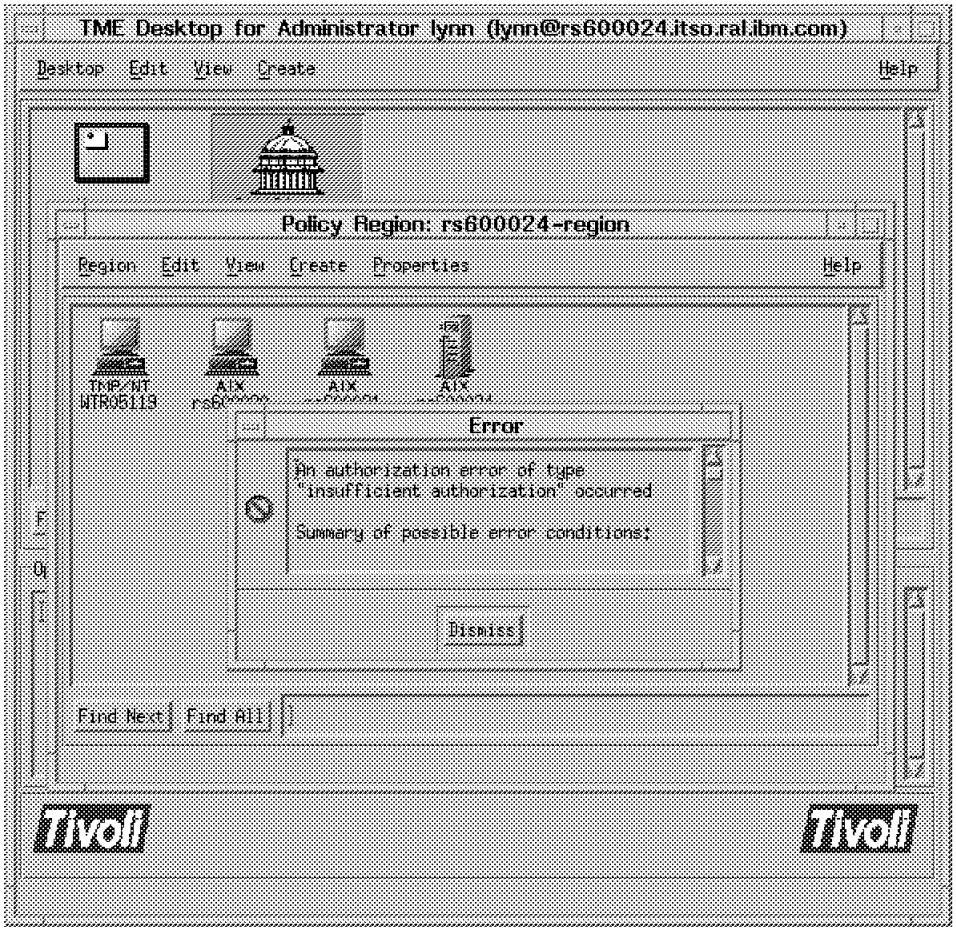


Figure 74. Insufficient Authorization to Create a Task Library

Now remove lynn′s *user* authority at the TMR level. This is in preparation for the example we will show later when lynn tries to execute a task. However, remember that if lynn has no authority at the TMR level, lynn′s Desktop can only be started on the TME Server, not on a Client.

To create the task library we need to use a suitably qualified administrator ID, for example:

1. From root′s desktop, double-click on the policy region icon, and then select **Create => Task Library** from the menu bar. Enter the name for the new task library (see Figure 75 on page 91) and select **Create & Close**.

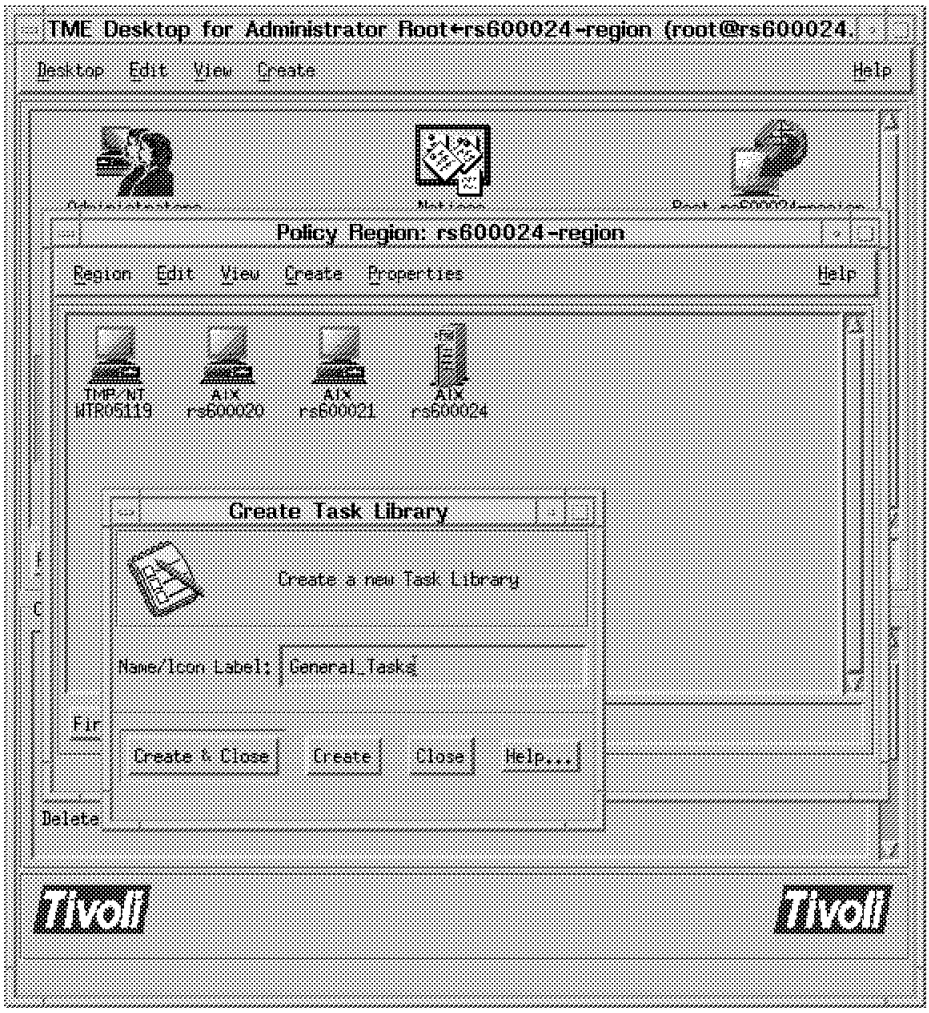


Figure 75. Creating a New Task Library

2. On lynn's desktop, the Task Library **General_Tasks** will appear under policy region **rs600024-region** panel and lynn can now add tasks to it by opening the Task Library icon and selecting **Create => Task** from the menu bar.

4.2.3 Create a Task

The next step is for lynn to create a task. The first task we use as an example will run the df command on target systems to show file system statistics. Some programs (notably shell scripts) can run on any operating system within the TMR. Other programs are system-specific. If the command or script to be executed is not the same on all platforms, separate entries will have to be made for each platform. In this case, the command will be the same on all platforms in the policy region, so we used the *Generic* system platform, as shown in Figure 76 on page 92. Note that you have to specify from where the program is to be retrieved by entering the managed node and file path where the executable can be found.

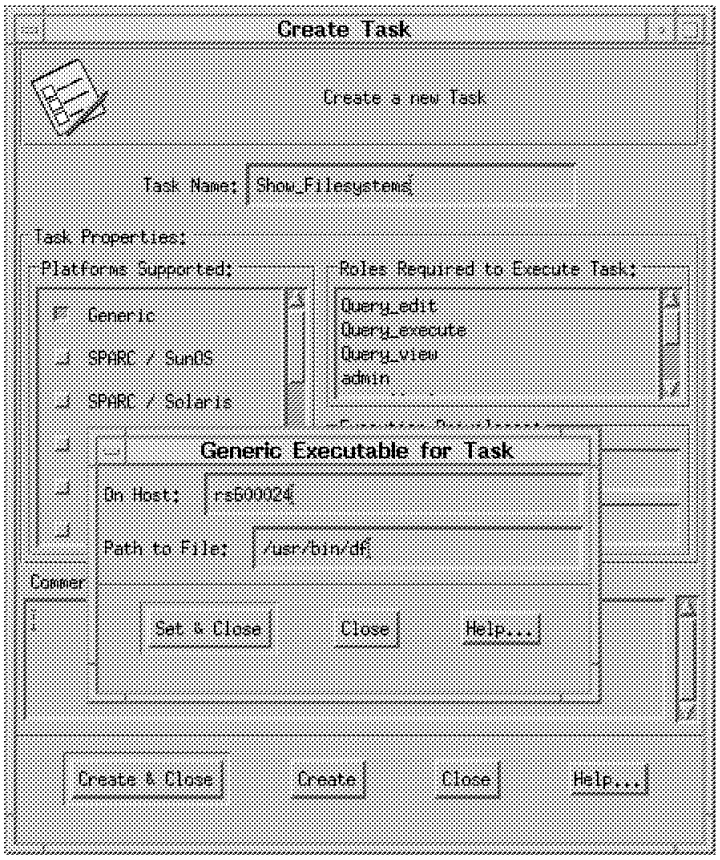


Figure 76. Creating a New Task Library

There are some additional details that you need to specify at this point:

1. Select a role from the list in the Roles Required to Execute Task field. This is where you control who has the ability to execute the task. In this case the task is a simple one and so we are not concerned with who can perform it. Therefore, the field was set to *user*, so that any Administrator with this level of authorization in the Policy Region can execute the task. Other tasks may be set to higher levels where they need to be restricted.
2. The Execution Privileges field can usually be left to default. This represents the ID under which the task will run on the managed node. The default is an asterisk (*****), meaning the ID of the user running the task.
3. You can optionally add comments to describe the task.
4. When all is complete, select **Create & Close**.

4.2.4 Create Task Libraries and Tasks from the Command Line

All of the preceding functions executed at the TME desktop can be run at the command line. This means the commands can also be set up in a shell script.

Here are all of the commands required to replicate the above sequence:

Create Administrator (wcrtadmin) lynn from root: To create tasks from the command line, lynn requires a TMR role of user as well as a policy region role of admin, whether running the command from the server or from a client.

```
wcrtadmin -l lynn@rs600024.itso.ral.ibm.com \
          -l lynn@rs600020.itso.ral.ibm.com \
          -r global,user \
          @PolicyRegion:rs600024-region,admin \
          -u lynn -g resident lynn
```

Create the Task Library (wcrttlib) from root

```
wcrttlib General_Tasks rs600024-region
```

Create Task (wcrttask) Show_Filesystems from lynn

```
wcrttask -t Show_Filesystems -l General_Tasks -r user \
         -c "Runs the df command to show Filesystems" \
         -i default rs600024 /usr/bin/df
```

4.2.5 Executing the Task

Now that the task has been created, administrator lynn can try to execute it. lynn does this by double-clicking on the task icon, or by clicking on it with the right mouse button and selecting **Execute Task** (see Figure 77 on page 94).

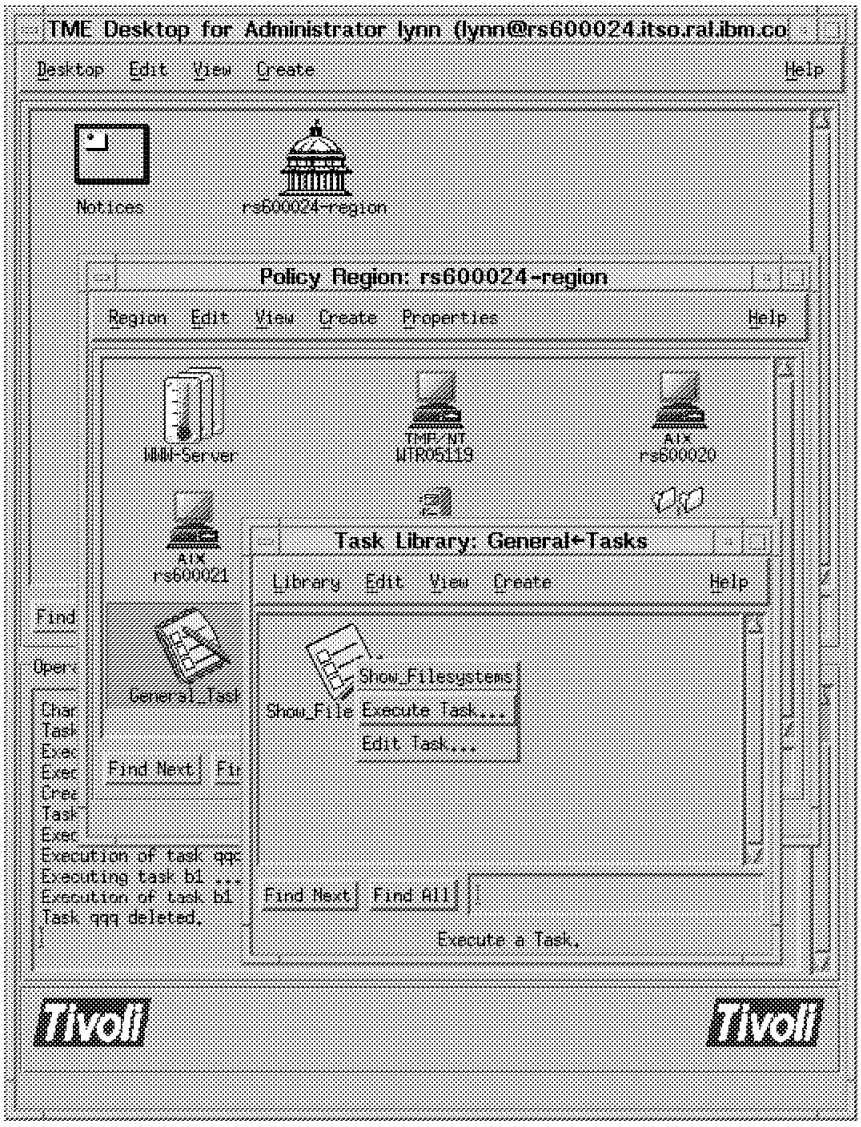


Figure 77. Executing a Task

The resulting dialog is shown in Figure 78 on page 95. Default values can be taken for the Execution Mode, Execution Parameters and Output Format, but you have to supply values for Output Destination and Task Endpoints. In this case we selected **Display on Desktop** as the Output Destination. We also selected one machine from the available Task Endpoints.

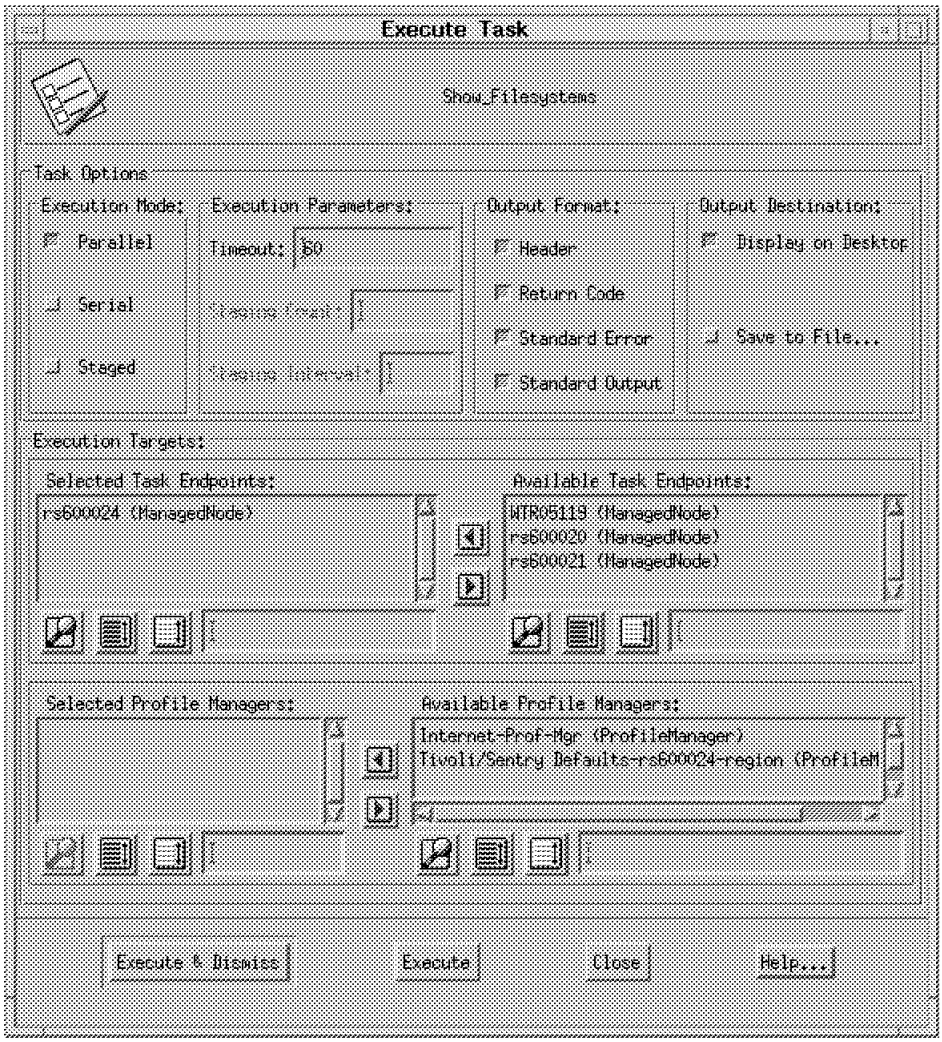


Figure 78. Execute Task Dialog

Next we click on **Execute** with the result shown in Figure 79.

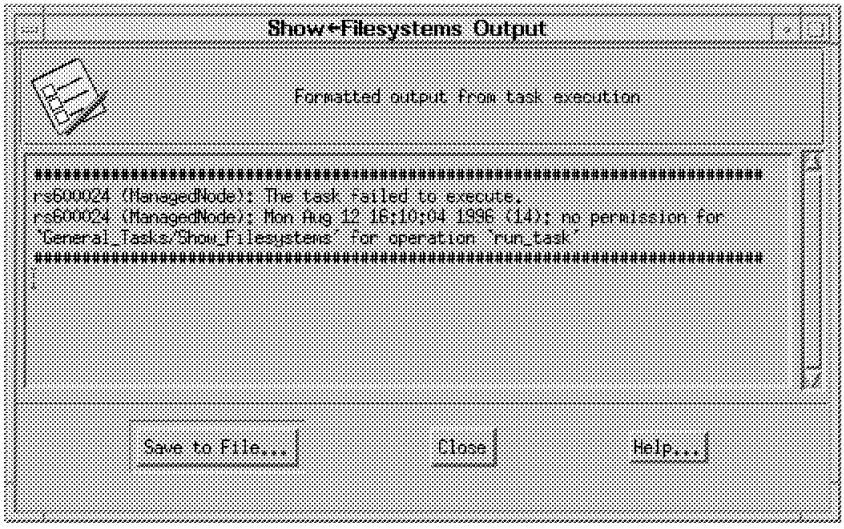


Figure 79. Task Fails to Execute Due to Insufficient Authorization

The fact that the task has failed may seem strange, given that it was administrator lynn who created it. The reason she cannot execute the task is because the task requires authorization of user and lynn has only admin. This highlights the fact that the authorization roles are not a hierarchy, but a set of individual roles. However, if we had not removed lynn's *user* authority on the TMR, the task could be executed by lynn.

From root's desktop, add *user* to lynn's resource roles in the policy region (see Figure 80).

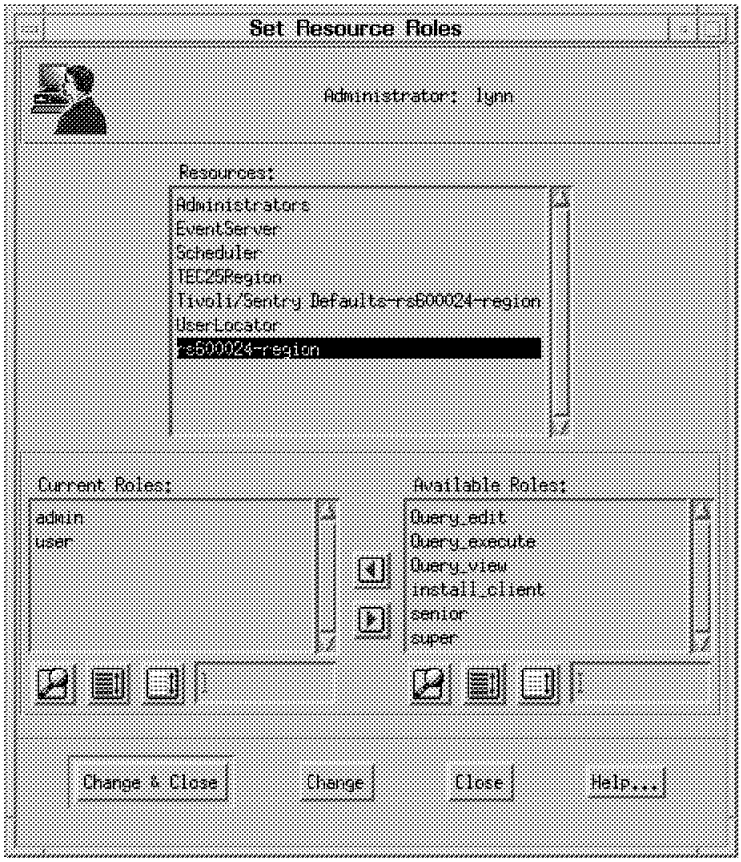


Figure 80. Modifying the Policy Region Resource Roles

Or from the command line:

```
wsetadmin -r @PolicyRegion:rs600024-region,admin:user lynn
```

Once lynn's desktop has been restarted, the task can now be executed, with the result shown in Figure 81 on page 97.

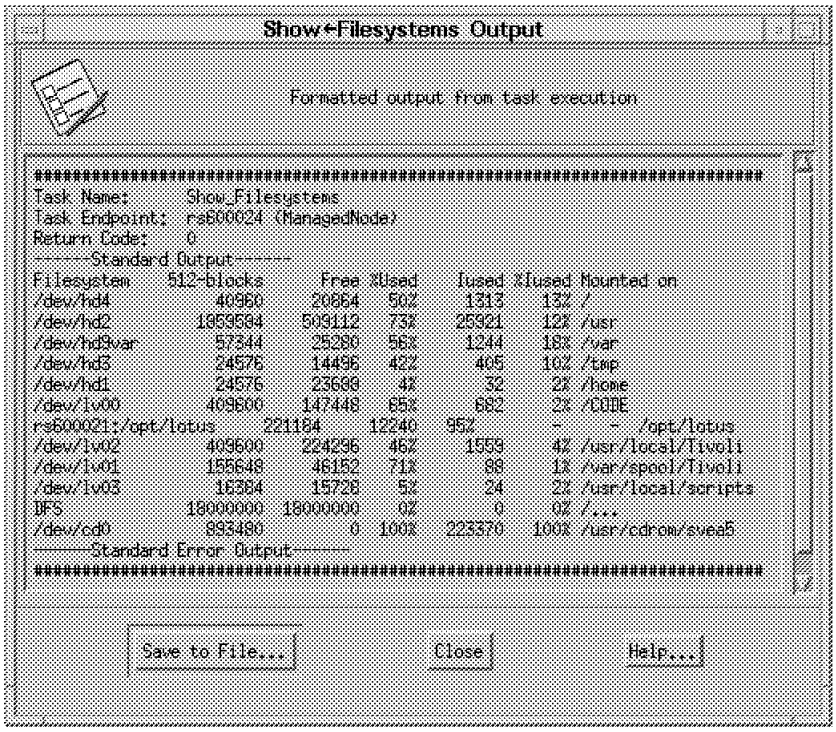


Figure 81. Successful Task Execution

Tasks can also be executed by dragging and dropping using the mouse. Hold down the left mouse button on the Show_Filesystem (Task) icon and drag it over one of the machines in the Policy Region window. Release the button to execute the task on that node.

Although we did not find this specifically documented, we discovered that we needed to add a TMR Role to administrator lynn to be able to use the drag and drop facility. We found that an authorization of user was sufficient.

4.2.6 Administrator Roles Needed to Execute a Task

The *Tivoli Platform User's Guide* shows the authorization roles required to perform all activities and can be referenced to gain a general understanding of the requirements. However, we found that sometimes access to tasks using a combination of authorization roles did not exactly give us the results we expected (as the previous example demonstrates).

The following table shows the results of trying to run a task with different levels of authorization for an administrator. There are three variables on the left side of the table:

1. The *role required to execute the task*. This is the role defined when the task was originally created (see Figure 76 on page 92).
2. The *role held by the administrator over the TMR*.
3. The *role held by the administrator over the Policy Region*.

The two columns on the right of the table show the result of trying to run the task with these authorization roles. A value of Y means that the task execution was successful, and N means that it failed.

Role required by Task	TMR role held by Administrator	Policy Region role held by Administrator	Execute from Client Node	Execute from Server Node
user	-	-	N/A	N
user	-	user	N/A	Y
user	-	admin	N/A	N
user	-	user,admin	N/A	Y
user	user	-	Y	Y
user	admin	-	N	N
user	user,admin	-	Y	Y
admin	-	user	N/A	N
admin	-	admin	N/A	Y
admin	user	-	N	N
admin	user	admin	Y	Y
admin	admin	-	Y	Y
admin	admin	user	Y	Y

This table can be summarized as follows:

- Whatever the task requirement is set to, the same authorization must be set for the Administrator.
- The Administrator authorization can be at the Policy Region level or the TMR level.
- The admin level is not *higher* than the user level.
- A TMR role is required to start the desktop from a client.

4.2.7 Executing a Task with Arguments from the Command Line

Both of the tasks we have created so far are simple operations, with no requirement for arguments to be passed. How do we handle commands that require a command line argument?

If we are executing the task from the command line, the answer to this is straightforward. For example, the wruntask command is used to execute the Show_Filesystems task on managed node rs600024 from the command line:

```
wruntask -h rs600024 -t General_Tasks -l Show_Filesystems
```

If we want to pass an argument to the command, we simply define it using the -a flag. For example:

```
wruntask -h rs600024 -t General_Tasks -l Show_Filesystems -a /tmp -a /
```

This command passes two arguments to the df command. These are the names of the two filesystems for which to display statistics. The output from this command will appear as shown in Figure 82 on page 99.

```
#####################################################
Task Name:      Show_Filesystems
Task Endpoint:  rs600024 (ManagedNode)
Return Code:    0
------Standard Output------
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd3           24576     14504   41%      403    10% /tmp
/dev/hd4           40960     20872   50%     1313    13% /
------Standard Error Output------
#####################################################
```

Figure 82. Result of the Show_Filesystems Task with Arguments Supplied

See the *Tivoli Platform Reference Guide* or the man pages for all of the options that can be set on the wruntask command.

4.2.8 Executing a Task with Arguments from the Desktop

To execute a task from the desktop, where the task requires arguments, we must use the Task Library Language (TLL). This is a more advanced topic than simple task creation, but it is not as complex as it may appear. We show an example of using TLL in 15.1, “Using the Task Library Language (TLL)” on page 455.

Part 2. Deployment Applications

© Copyright IBM Corp. 1996

Chapter 5. Tivoli/Courier

A primary issue in Information Technology is deployment management. It is even more complicated by large, distributed networks with multiple operating systems and applications. A successful deployment management solution should be able to:

- Configure machines running different operating systems, which are geographically dispersed
- Install software in a timely and efficient manner
- Maintain software by performing frequent updates
- Having the capability to remove software updates when required
- Monitor the software and data to ensure that it is synchronized with other systems

Existing deployment management solutions often fail to reduce the complexity of heterogeneous and distributed environments.

Courier provides a means of managing and distributing software across a multi-platform network including the machines listed below:

- UNIX servers
- NetWare servers
- PCs running Windows
- Windows NT servers
- OS/2 servers and workstations
- MSDOS workstations

Courier gives the administrator a centralized software management capability to add new applications, update existing software with newer versions, and synchronize software on distributed systems.

5.1 Courier Features

In this section, we discuss features and capabilities of Courier.

5.1.1 File Packages

Courier uses the concept of file packages to distribute software from a source machine to target machines. A file package contains the *information* relating to the files and directories that will be distributed, not the actual files and directories. It also contains the necessary actions to be executed as part of the distribution. For example, you may want to mount a CD-ROM to a drive on the source machine before the distribution starts, because the files that you are going to distribute are in the CD-ROM.

Since the actual files are not contained in the file package, it is possible to change or update the files without changing the file package. The next time you distribute the file package it will be distributing the updated files. A file package is treated as a *profile* in Tivoli Management environment (TME).

© Copyright IBM Corp. 1996

A Courier File Package is therefore different from a *NetView DM Change File*.

Courier lets you take a *snapshot* of a file package to create a *file package block* or *fpblock*. The fpblock will contain all of the files and directories to be distributed and any configuration programs required. Once the fpblock is created, its content cannot be changed. This is similar to a NetView DM Change File.

Once the file package has been created, you can then create the fpblock from the command line only. Refer to the *Tivoli/Courier Reference Manual & User's Guide* for a detailed description of the commands to manipulate the fpblock.

The commands are as follows:

- **wcrtfpblock** to create an fpblock
- **wdistfpblock** to distribute the fpblock
- **wrmfpblock** to remove the fpblock
- **wcpfpblock** to copy the fpblock from one TME node to another

So when should you use a file package instead of an fpblock? A file package will allow you to dynamically change the files that you want to distribute without the need to rebuild an fpblock every time. An fpblock will ensure that once you have the file package, the contents will never change. This will enable you to distribute identical copies of the file package to all targets at any time.

If you have a slow network connection between servers, such as a WAN, and each server is connected by a LAN to many PC clients, you will distribute the fpblock from the controlling TMR to each server. Then each server will manage the installation of the fpblock to its PC managed nodes increasing the performance of the distribution.

5.1.1.1 Source Host

The source host is where files and directories are kept for distribution. It must be a UNIX managed node. A file package can only have one source host. The source host can belong to any Tivoli Management Region (TMR). For example, you can have your source host in one TMR and have your target machines in some other TMR.

5.1.2 Nested File Packages

A file package can contain other file packages in addition to having files and directories. When a file package is being referred by another file package, then it is called a nested file package.

When you distribute a file package that has files, directories and nested file packages, you are distributing files and directories referred by the main file package and nested file packages.

If you distribute a nested file package that does not have any file packages included in it, only the files and directories referred by the nested file package will be distributed.

It is also possible to create a file package that does not distribute any files or directories, but only performs actions.

5.1.3 File Package Operations

After you create a file package, you can distribute the file package to its subscribers. Subscribers can be managed nodes, PC managed nodes, and other profile managers. When you distribute the file package, the files specified in the file package are actually distributed to the target systems.

5.1.3.1 Distribute

The Distribute operation will distribute the file package on the specified targets. It will also trigger any before and after distribution programs contained in the file package.

Courier supports only the push method of distributing file packages. To pull a file package, you need to have Tivoli/UserLink installed on a Windows, Windows 95 or Windows NT and on the managed node. The pull method is not supported on DOS, OS/2 or NetWare machines.

We did not investigate the usage of Tivoli/UserLink in our project. For more information, see the *Tivoli/UserLink User's Guide*.

The distribution operation is similar to the NetView DM combined operation of *send* and *install.* In NetView DM you can also specify removability=yes to save a backup copy.

5.1.3.2 Remove

The remove operation will remove the file package on the specified targets. It will also trigger any Before and After removing programs contained in the file package.

The remove operation is similar to the NetView DM remove operation. In NetView DM, the remove operation will restore the backup copy made during the installation step.

5.1.3.3 Commit

The commit operation will run the commit program contained in the file package on the specified target once the file package has been distributed.

The commit operation is similar to the NetView DM *accept*. In NetView DM, the accept operation will erase the backup copy of the package making the changes permanent.

5.1.4 Configuration Programs

When distributing a file package to a target node, it is possible to run configuration programs on the target node:

- Before or after distributing software
- During a commit operation
- Before or after removing file packages
- If an error occurs on a target that stops the distribution or removal

The configuration programs could reside on the source host or on the target node. If they reside on the source host, Courier copies them to a temporary area in the target node before it runs. You do not have to include them as files to be distributed in the package. After the execution, Courier will remove them from

the target node. In addition, you can also execute configuration programs on the source host:

- Before distributing software to a target
- After distributing software to a target

The configuration programs that need to run on the source host must reside on the particular source host.

On UNIX, you can run any executable program (for example, shell script, C program, and Perl script). On NetWare servers, you can run NetWare loadable modules (NLMs) and .NCF files. On PCs, you can run executable programs written as .BAT, .EXE, .COM or .CMD files.

These configuration programs are similar to the NetView DM Pre and Post scripts.

You can create and update file packages using the TME desktop. Using the TME desktop, you can define the following configuration programs:

- Before distribution on target node
- After distribution on target node
- Before removal on target node
- During commit on target node

If you want to define other configuration programs, you have to export the file package to a text file, insert the parameters to define required programs, and then import the text file to the file package. Even after updating the file package using this method, you would not be able to see the additional information using the TME desktop, but these additional configuration programs will be executed.

5.1.5 Import/Export File Packages

You can export a file package to a text file and import a text file to a file package in order to modify and customize the file package. This is useful when you have a long list of files or nested file packages to distribute. It is also useful when you want to create file packages with options that are not supported through the dialog.

This means that you can create a file package using the dialog. But if you want to have a program in the package run in case an error occurred during the distribution, then the dialog does not allow you to specify such a program. Therefore, you need to export the file package to a file, edit the file to add the program that you want to run to the appropriate keyword (for example, unix_on_error_prog_path) and then import the file package.

You can also import a standard *component description file* (CDF) into a file package using the *Application Management Specification* (AMS) functionality of Courier. AMS is produced by the *Desktop Management Task Force* (DMTF). The AMS enables a developer to divide an application into components. Each component contains a set of files that contain information about the application. These application components are described by the developer using a set of files, each of which is known as a component description file.

During our residency we did not use CDFs to update file packages.

Note: To import a text file or a CDF to a file package, the file package must exist within a profile manager. When you import a text file or CDF, the file package will be updated.

The Tivoli/Courier import/export feature is similar to the NetView DM import/export a profile feature.

5.1.6 WAN-Smart Capabilities

Tivoli/Courier uses WAN-smart capabilities that reduce network traffic. The capabilities include parallel distribution, fan-out and network bandwidth tuning.

5.1.6.1 Multiplexed Distribution

Tivoli/Courier performs distribution in parallel when distributing software to multiple target machines. If endpoints reside in a remote TMR network, Tivoli/Courier automatically makes the initial distribution to the TME server on the target TMR. Then from that server a local distribution takes place on the target machines. You can fine tune this feature to control the number of simultaneous parallel distributions that should be initiated from each fan-out server.

In order to take full advantage of those capabilities you need to set *repeaters* in your network.

5.1.6.2 Network Bandwidth Tuning

In addition to using repeaters to conserve bandwidth, you can fine tune various network parameters to control the percentage of network bandwidth used during a distribution.

You can set values for repeater parameters such as disk space to use, maximum amount of data to send, and maximum memory to be used by issuing the `wrpt` command. Refer to the *Tivoli Management Platform Reference Manual* for a detailed description of the command.

5.1.7 Security

Tivoli/Courier functions within the TME authorization roles. To perform operations within Tivoli/Courier, you need the required authorization role for the task. To perform system administration operations within the TME, you need to be a Tivoli administrator. Depending on what operations you are required to perform, you can have one or more of the following roles:

- Super
- Senior
- Admin
- User
- Install-product

5.1.7.1 Setting File Package Profiles

The following table lists the roles required to set up file package profiles:

Table 8. Setting File Package Profiles		
Operations	**Context**	**Required Role**
Install Tivoli/courier	Desktop view for root	**super** or **install-product**
Create or Clone a file package	Profile manager	**super** or **senior**
View a file package	File Package	**super**,**user**,**admin** or **senior**
Subscribe resources to a profile manager	Profile manager's policy regions AND Subscriber's policy region	**super**,**admin** or **senior**

5.1.7.2 Defining and Deleting File Packages

The following table lists the roles required to define a file package by setting its properties and to view a file package's configuration information:

Table 9. Defining and Deleting File Packages		
Operations	**Context**	**Required Role**
Export a file package definition	File package	**super**,**user**,**admin** or **senior**
Set or edit file package properties	File package	**super** or **senior**
Import a file package definition or a CDF	File package	**super** or **senior**
Delete a file package	File package	**super** or **senior**

5.1.7.3 Performing File Package Operations

The following table lists the roles required to distribute a file package:

Table 10. Performing File Package Operations		
Operations	**Context**	**Required Role**
Distribute a file package	Profile manager's policy region AND Subscriber's policy region	**super**,**admin** or **senior**
Schedule a file package distribution	Profile manager's policy region AND Subscriber's policy region	**super**,**admin** or **senior**
Calculate the size of a file package	Profile manager's policy region	**super**,**admin** or **senior**
Remove a file package from a subscriber	File package	**super**,**admin** or **senior**

5.1.8 Resources

Figure 83 on page 109 introduces all of the resources that are used by Tivoli/Courier:

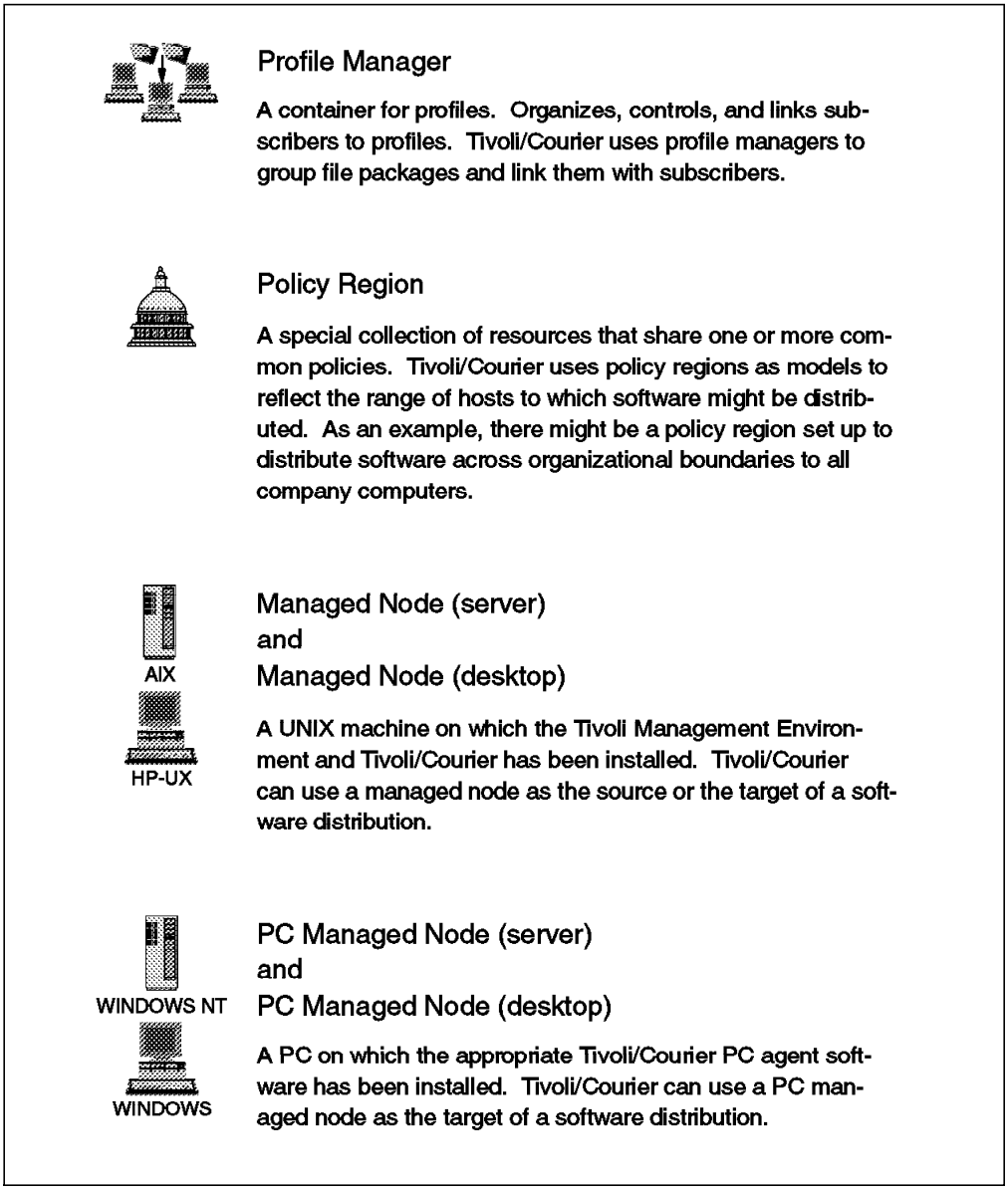


Figure 83. Tivoli/Courier Resources

5.2 Installation

You can install the Courier application from the TME desktop or command line. In our setup, we used the TME desktop to install Courier.

5.2.1 Planning the Installation

Before you install Courier, you need to have Tivoli Management Platform installed. Courier needs to be installed only on TMR servers and any UNIX managed node that you are going to use as a Courier repeater. The installation process for both servers and the repeaters are the same.

We installed Tivoli/Courier on TME servers rs600011 and rs60008.

The installation process for Courier is similar to any other Tivoli product. It is described in 2.5, “Product Installation” on page 40.

If you install Courier on a TMR server, you need to update managed resource types of policy regions where you want to create Courier file packages. Also, you need to update the notice group subscriptions for any administrators, if they want to receive Courier notices.

5.3 Configuring Courier Distribution Environment

After having installed Tivoli/Courier on the TME servers it is necessary to configure the distribution environment. Please refer to 2.1, “TME Management Concept” on page 17 to view the Tivoli Management Region (TMR) structure used in our project.

The following sections take you through the steps necessary to define a software distribution environment.

5.3.1 Policy Regions

The policy regions Prod and SD are the default policy regions created when the TME servers were installed.

We also created a policy region called SoftDist.

Note: It is important to clarify that it is not necessary to organize your environment in the same fashion as we did. You do not need to create the additional policy region SoftDist because you can use the default ones.

However, we created the new policy region to logically group all of the objects related to software distribution.

Another reason could be that if the TMRs in your organization have the policy regions organized by department, you might find it useful to maintain file packages in a separate policy region. In this way, you can distribute software across departmental boundaries to all company computers.

Please refer to 1.1.3, “Policy Regions” on page 5 on how to create and define policy regions.

5.3.2 Network Distribution Environment

Once the TME servers have been installed, you also need to install TME on the Managed Node and the PC Managed Node. Please refer to 2.4, “Installing the TME Management Platform” on page 23 for a detailed description on how to install TME on those nodes.

In our scenario we have two TMRs connected with a two-way connection:

- Prod with server rs600011
- SD with server rs60008

In both TMRs there are various nodes defined, since other Tivoli projects were taking place at the same time at the ITSO. For software distribution purposes, we use only server rs600011 from TMR Prod and all the other nodes from TMR SD.

Therefore, the following scenario will be used for software distribution:

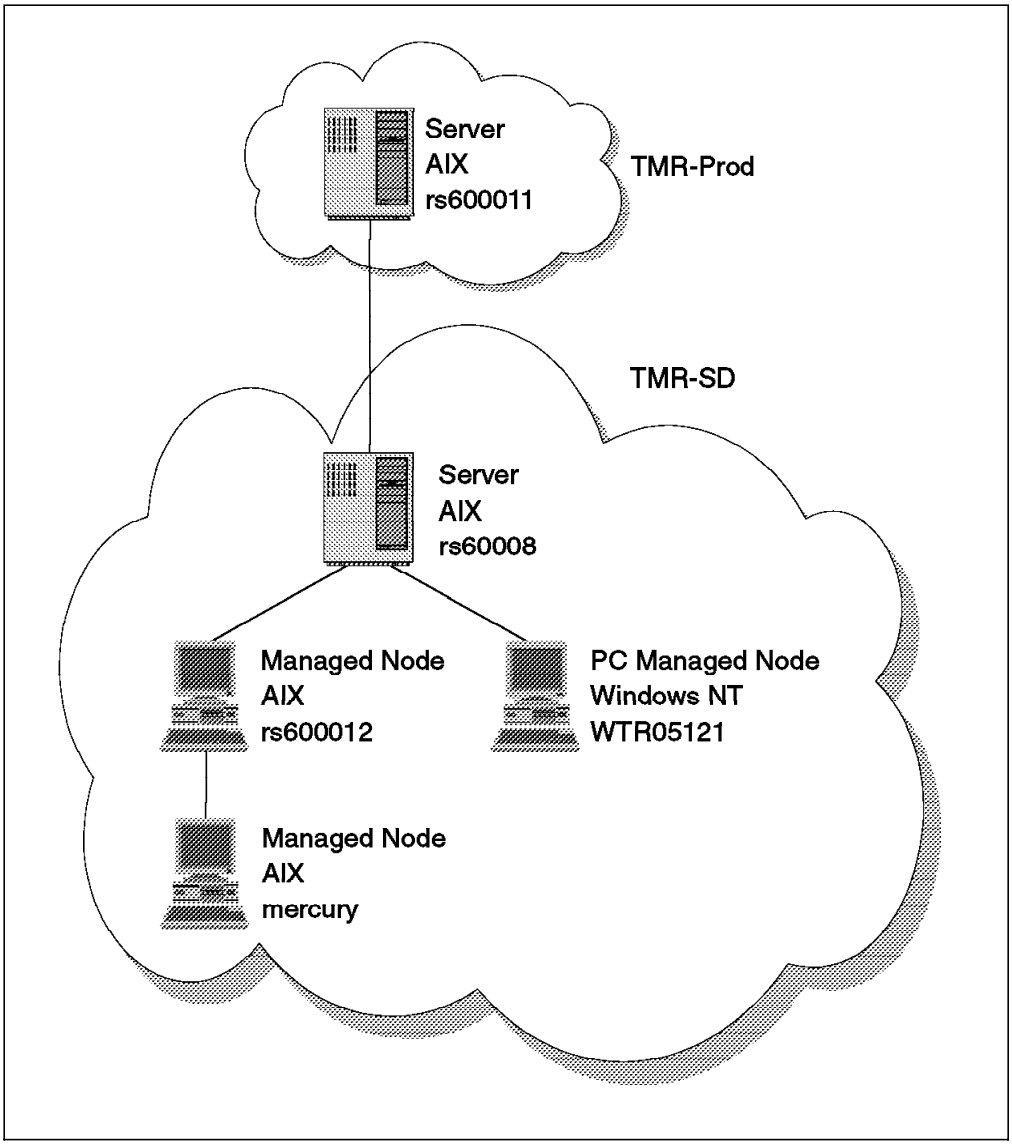


Figure 84. Software Distribution Scenario

5.3.2.1 Configuring Repeaters

Repeaters are machines that in a TMR receive and distribute data in parallel with other machines. By default a repeater site is created on the TME server when you install the platform. Therefore, rs600011 and rs60008 are already defined as repeaters.

To facilitate highly efficient transmission of data across a variety of networks, the TME provides the Multiplexed Distribution (MDist) service. MDist is used to establish a hierarchy of distribution repeater servers, each of which is capable of distributing data in parallel with a large number of clients.

For large TMRs or TMRs with slow links, additional repeaters can be defined to increase the efficiency of data distribution.

A repeater is known as an intermediate node in NetView DM.

In our environment we defined rs600012 as the repeater in TMR SD.

To define a repeater you have to:

1. Determine which machine is defined as the repeater by entering the command `wrpt` from any managed node in the TMR. The output of the command is:

```
rs600011 [1]      wd-  [default]
rs60008 [1]       wd-  [default]
```

 These are the default repeaters created when TME was installed on the servers.

2. List the machines in the range of rs600011 and rs60008 repeaters by entering the command `odadmin odlist` from any managed node in the TMR. The output of the command is:

```
Region     Disp  Flags  Port        IPaddr   Hostname(s)
1525396064  1    ct-    94     9.24.104.123  rs600011.itso.ral.ibm.com
           12    ct-    94     9.24.104.109  rs600010.itso.ral.ibm.com
           15    ct-    94     9.24.104.152  venus.itso.ral.ibm.com
           17    ct-    94     9.24.104.249  rs600019.itso.ral.ibm.com
           18    ct-    94      9.24.105.31  sun.itso.ral.ibm.com
           19    ct-    94      9.24.105.32  hp.itso.ral.ibm.com
2071979149  1    ct-    94      9.24.104.30  rs60008.itso.ral.ibm.com
            3    ct-    94      9.24.104.27  rs60004.itso.ral.ibm.com
            8    ct-    94     9.24.104.124  rs600012.itso.ral.ibm.com
            9    ct-    94     9.24.104.117  mercury.itso.ral.ibm.com
```

 Please note that the output of the command shows only the *managed node*; it does not show the PC managed node.

3. Identify in the output list the value for the field Disp for the host or range of hosts that the repeater must manage. In our case, we wanted to define rs600012 as the repeater and define mercury as the host to be managed. The Disp value for mercury is 9.

4. Therefore, to define the repeater, you need to enter the command:

```
wrpt -n rs600012 range=9 always
```

 Note: The option `always` forces the distribution to go through the repeater although the repeater has only one client. By default, if a repeater has only one client, a distribution to that client goes directly to the client from the TME server, bypassing the repeater. Since rs600012 has only one client defined, we used the option `always`.

5. Enter the command `wrpt` from any managed node in the TMR to verify the correct definition of the repeater. Now the output will be:

```
rs600011 [1]      wd-  [default]
rs60008 [1]       wd-  [default]
rs600012 [8]      --a [9]
```

Refer to the *Tivoli/Courier User's Guide* and *Tivoli Management Platform Reference Manual* for a complete description of the commands `odadmin odlist` and `wrpt`.

5.3.3 Create Profile Managers

Now you need to create several Profile Managers inside the SoftDist policy region. These Profile Managers will be the folder of the nodes and file packages.

5.3.3.1 Create Profile Managers to Group Nodes and File Packages

We now define Profile Managers to contain the various type of machines and file packages.

These Profile Managers are folders that will then be populated with nodes and file packages. Therefore, you can define and organize them in the way that would best suit your environment.

We created Profile Managers to group nodes such as AIX and Windows NT. We also created Profile Managers to group file packages for AIX and Windows NT. It is a good approach to create a Profile Manager for each platform type, since file packages often contain platform-specific data.

1. From the desktop double-click on the **SoftDist** policy region. Then select **Create** = > **ProfileManager** and fill in the name of the Profile Manager that you want to create:



Figure 85. Create Profile Manager AIXMachs

2. Repeat the previous step for as many Profile Managers you intend to create.
3. After we created all of the Profile Managers, the following icons were displayed in our policy region SoftDist:

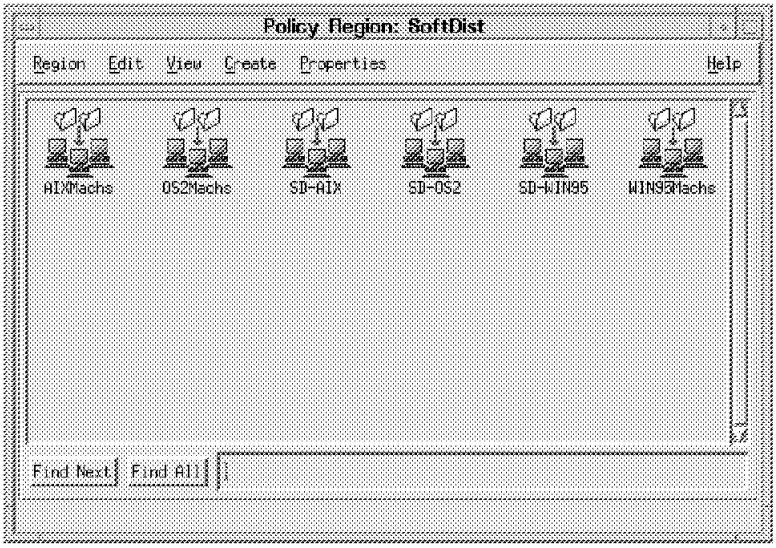


Figure 86. Policy Region SoftDist

5.3.3.2 Populate Profile Managers with Subscribers

For each Profile Manager that represents a different type of machine, you need to add the nodes defined in your TMR that you want to target with software distribution. These nodes are known as *subscribers* to the Profile Manager.

1. From the desktop double-click on the **SoftDist** Policy Region and to display all of the Profile Managers previously created as shown in Figure 86.
2. Now double-click on the **AIXMachs** icon. The panel will not show any subscribers at this stage.
3. To subscribe the AIX managed node, select **Profile Manager** = > **Subscribers**. The TME displays the following dialog:

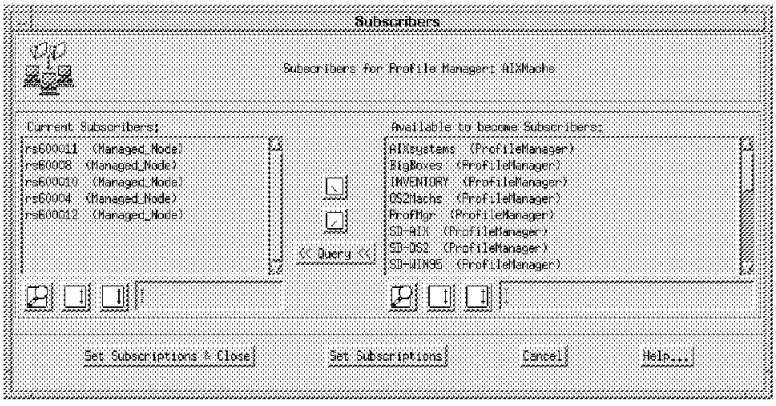


Figure 87. Subscribers to AIXMachs

All of the managed nodes and PC managed nodes available in the TMR are listed in the panel.

4. Select all of the AIX managed nodes from the Available to become Subscribers list. Then click on the left mouse button to move the managed nodes to the Current Subscribers list. Then select the **Set Subscriptions & Close** button to set and return to the Profile Manager window.

Now the Profile Manager window will have listed all of the subscribers that you have previously selected:

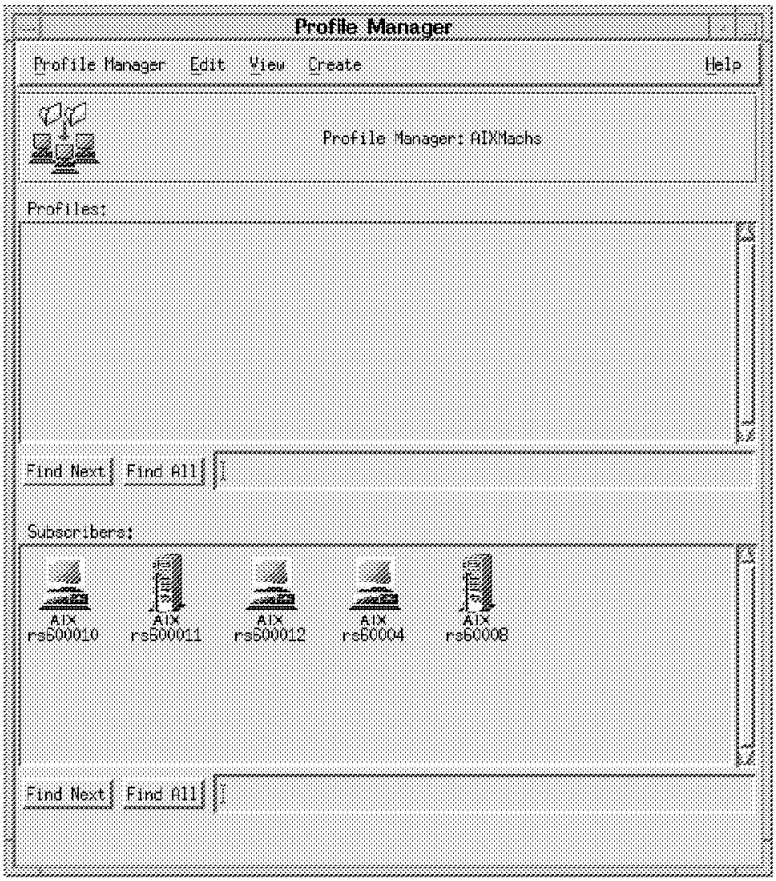


Figure 88. Profile Manager AIXMachs

5. Repeat the previous steps for all of the other Profile Managers to fully populate the Profile Managers with all of the nodes in your network.

You could also drag subscriber icons onto the Profile Manager′s icon view from the managed node and PC managed node available in the TMR.

5.3.3.3 Populate Profile Managers of File Packages

For each Profile Manager that represents a different type of file package (for example, AIX, OS/2 or Windows 95), you need to create the folders for each file package that you intend to distribute. Creating the file package only creates the object in which you include data to be distributed.

In our scenario, we show Profile Manager SD-AIX that will contain file packages INed and AIX_mqm^2.1.

Once the Profile Package has been populated with all of the file packages, you will also need to subscribe the Profile Manager′s subscriber previously created in 5.3.3.2, “Populate Profile Managers with Subscribers” on page 114.

1. From the desktop double-click on the **SoftDist** Policy Region. This will display with all of the Profile Managers previously created (see Figure 86 on page 114).

2. Now double-click on the **SD-AIX** icon. The panel will not show any file packages or subscribers at this stage.
3. To create a file package select **Create** = > **Profile**. The TME displays the following dialog:

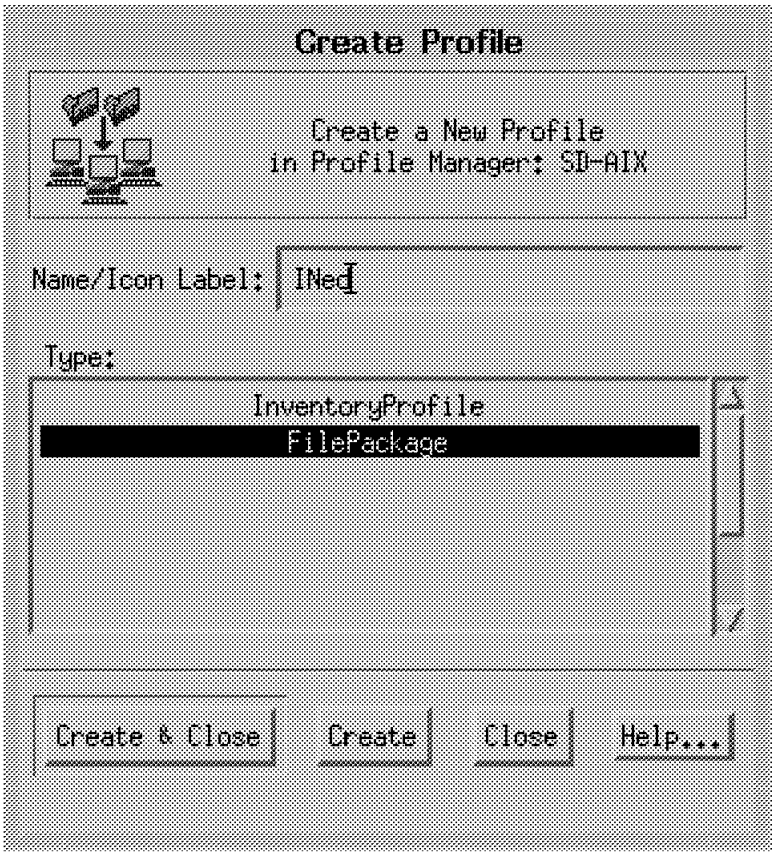


Figure 89. Create File Package INed in Profile SD-AIX

Enter the name of the File Package to create in the Name/Icon Label field and then select **FilePackage** from the Type scrolling list. Then select the **Create & Close** button to create the file package and return to the Profile Manager window.

Note: The *Tivoli/Courier User's Guide* recommends that you include the name and version of the file package, separated by the character ^ in the profile name. Using this format enables Tivoli/Userlink to distinguish between multiple versions of the same package. We did create the MQSeries package using this naming convention but did not investigate the usage of Tivoli/UserLink.

4. Repeat the previous steps for all of the other file packages that you intend to create.

Now the Profile Manager window will have listed all of the file packages created:

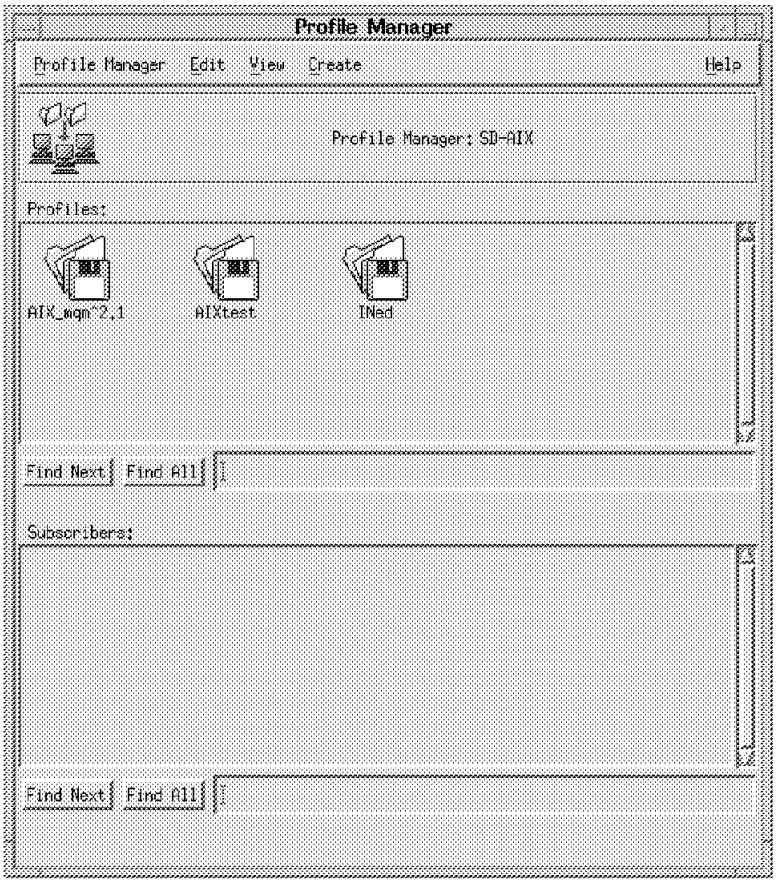


Figure 90. SD-AIX Profile Manager

5. Now that all of the file packages are created for Profile Manager SD-AIX, you need to create subscribers for the Profile Managers in order to perform software distribution functions.

 To do so, you can drag the Profile Manager icon that describes the AIXMachs from the policy region. Then drop it into the icon of the Profile Manager SD-AIX or perform the steps described in 5.3.3.2, “Populate Profile Managers with Subscribers” on page 114. This section details subscribing the platform-specific Profile Managers (in our case, AIXMachs).

6. After subscribing the AIXMachs Profile Manager the SD-AIX Profile Manager will look as follows:

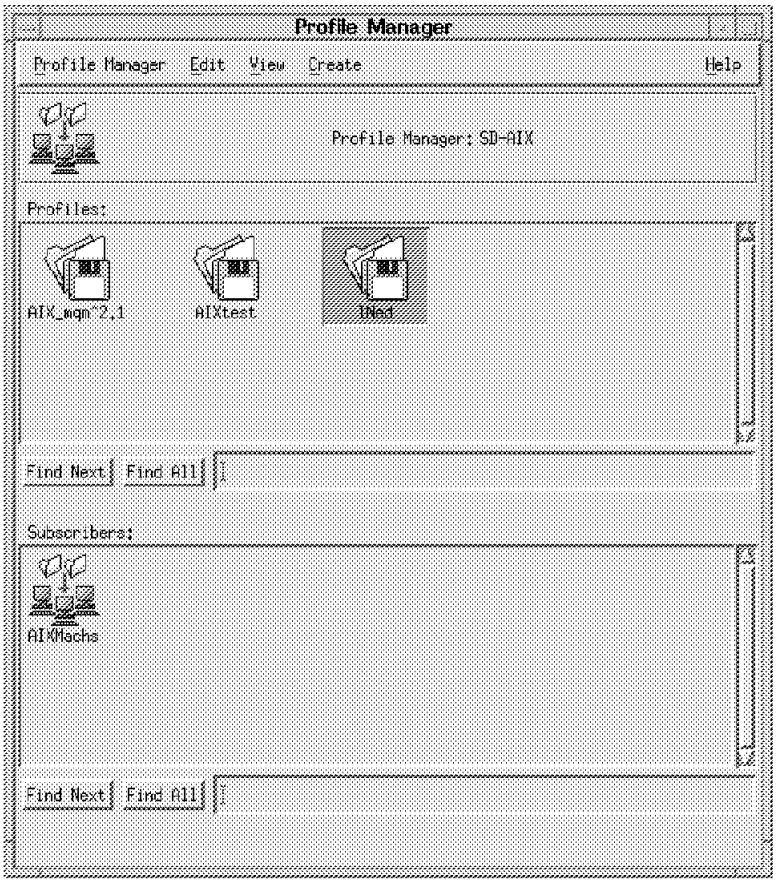


Figure 91. SD-AIX Profile Manager

To summarize we can say that a file package is a TME resource that describes a set of files and directories to be distributed. File packages are created in profile managers, which in turn reside in policy regions.

Managed nodes or PC managed nodes and profile managers qualify as subscribers. Subscribers need not be in the same policy region or even the same TMR as the profile manager that contains the file package to be distributed.

The following picture shows the relationship between those elements:

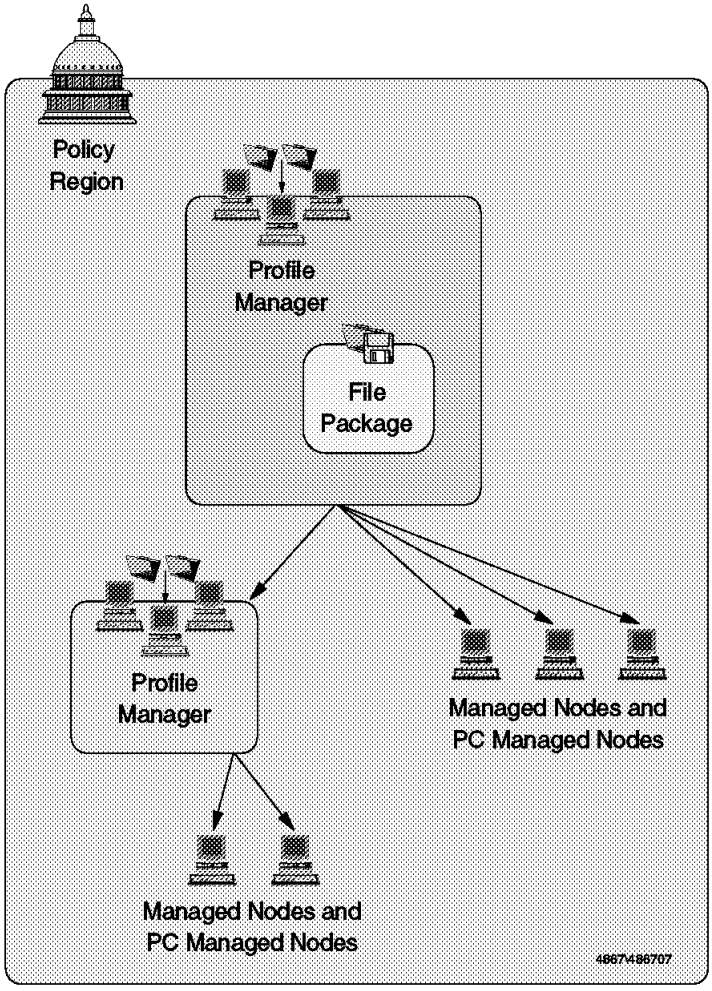


Figure 92. Tivoli/Courier Elements

The configuration of the Courier distribution environment is now complete.

5.4 Packaging and Distributing Software

After having defined all of the profile managers as described in 5.3.3, “Create Profile Managers” on page 113, it is now necessary to create and define properties and program options for the file packages.

This chapter describes how to:

- Plan the installation
- Create the staging area for the source files
- Define the file package properties, such as which files will be distributed and what program options will run on the target machines
- Create the program options
- Log information and notification behavior about the file package distribution
- How to distribute the file package to the target machines

We show three scenarios of file packages:

- INed Editor for AIX
- MQSeries for AIX
- MQSeries for NT

5.4.1 Create the INed File Package

The INed Editor package is available in installp image format.

5.4.1.1 Create the Staging Area for the Source Files

We used rs600011 as our source machine, so we copied the INed installp image into:

```
/courier/images/aix/INed.usr.3.2.0.0
```

Courier will always read from the source machine when distributing a file package. Therefore, if the software is available on CD-ROM, to avoid having to mount a CD-ROM drive every time you want to distribute the file package, it is recommended that you copy the image to a staging area.

5.4.1.2 Create the File Package Properties

You can define file package properties using the desktop, the command line, the export/import capability or importing a component description file. Our scenario uses the desktop.

1. From the SoftDist policy region double-click on the **SD-AIX** profile manager previously created in 5.3.3.1, “Create Profile Managers to Group Nodes and File Packages” on page 113.

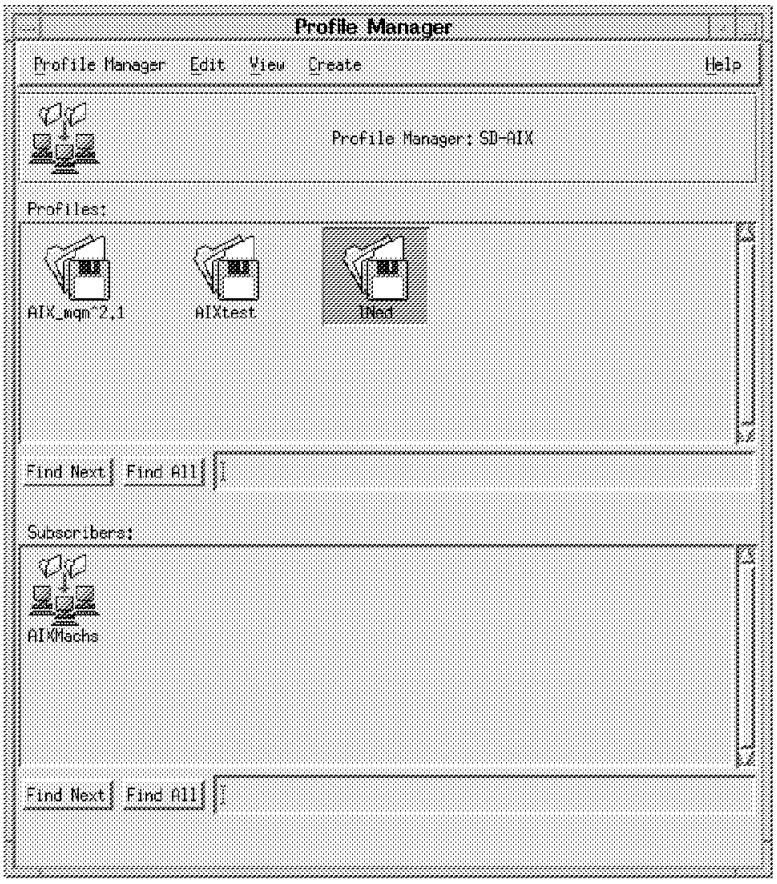


Figure 93. Profile Manager (SD-AIX)

2. Double-click on the **INed** file package. The TME displays the following dialog:

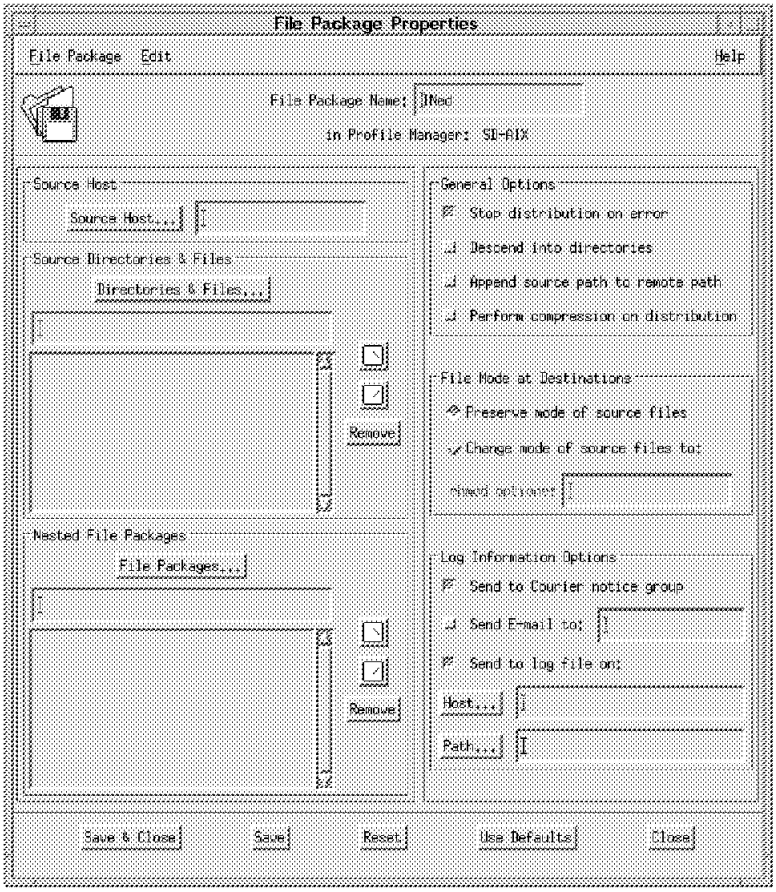


Figure 94. File Package Properties (INed - before)

The File Package Properties window shows the file package name, the directories and files and the nested file packages to be included.

3. Set the Source Host field of the file package.

 Click on the **Source Host...** field to identify the name of the host on which the file package source files reside. In our case, the host is rs600011.

 Source Host

 Only UNIX managed nodes are supported as source hosts.

4. Set the Source Directories & Files field of the file package.

 Click on **Directories & Files...** to identify the full path of the source files. In our case, the path is:

   ```
   /courier/images/aix/INed.usr.3.2.0.0
   ```

5. Set the Log Information Options field to control the logging activity.

 Select the **Send to Courier notice group** check box to have Courier post a notice, which includes an indication of success or failure of the operation for each target, to the Courier notice group when a file package operation occurs.

 Select the **Send to log file on:** check box to have Courier place log information in the specified file when a file package operation occurs. An entry is made to the log each time a file package is distributed, committed or removed.

In our project, we defined the Host... as rs600011 and the Path... as /courier/logs/aix/INed.log

Log

You should set the **Send to log file on** check box. Otherwise, vital information regarding the distribution operations will never be logged.

6. We then left the default in all of the other boxes.

 The File Package Properties window will now look as follows:

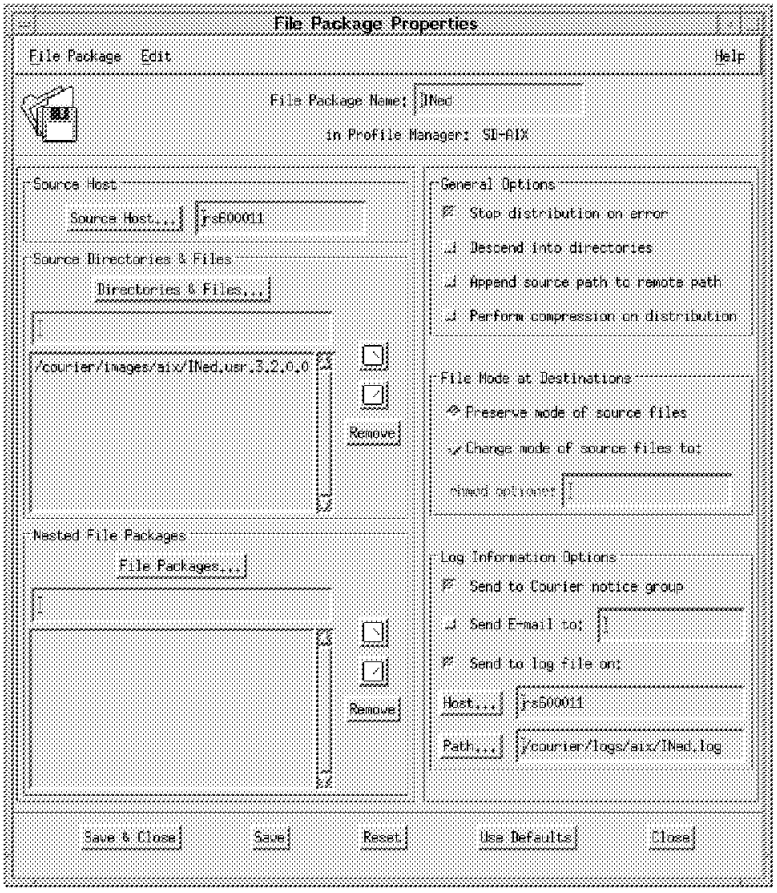


Figure 95. File Package Properties (INed - after)

5.4.1.3 Create the File Package Platform-Specific Options

After defining the file package properties, define the platform-specific options of the file package. In our example we defined UNIX file package options.

1. From the File Package Properties window shown in Figure 94 on page 122, select **Edit => Platform-Specific Options => UNIX Options...** to display the following panel:

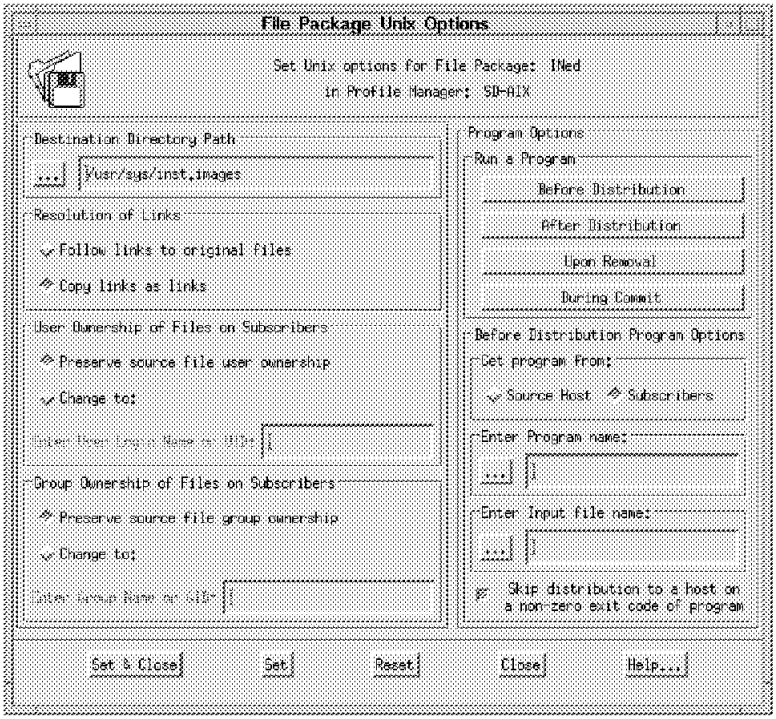


Figure 96. File Package UNIX Options (INed Before)

2. Set the Destination Directory Path to the directory full path of the target to which the file package is distributed. In our case we specified:

 `/usr/sys/inst.images`

3. Set the Resolution of Links to determine how symbolic links are handled during distribution and removal of file packages.

 Select the **Copy links as links** radio button to create symbolic links to the original files at the destination.

4. Set the Program Options using the Run a Program buttons.

 These buttons display program options that enable you to run programs (executable modules) or procedures, such as, C programs, shell scripts, and Perl scripts, on subscribers before or after distributing, removing or committing a file package.

 The INed package needs the following program to be run:

 After Distribution This is a script to install the package.

 Upon Removal This is a script to remove the package.

 During Commit This is a script to commit the package.

5. Figure 96 shows the Before Distribution Program option, which in our case does not need to be set. This is because we do not have to run any program on the target before distribution.

6. When you select the **After Distribution** push button, the following dialog is displayed:

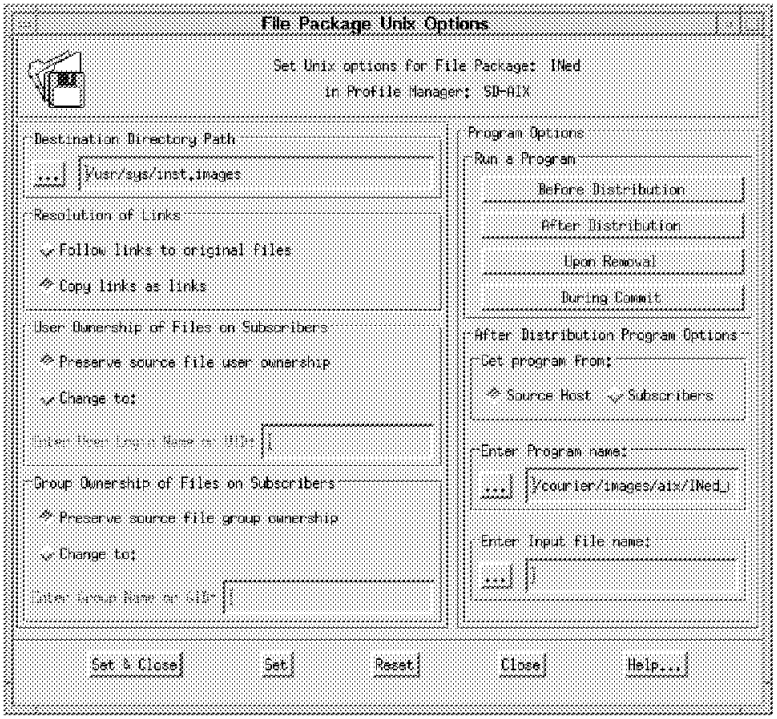


Figure 97. File Package UNIX Options (INed After)

7. Set the Get Program from field to specify the location of the program or procedure.

 Select the **Source Host** radio button to instruct Courier to copy the After Distribution program from the source host to a temporary file on each subscriber. It will then run the programs on each subscriber after the distribution. Next it removes each temporary file from each subscriber upon completion.

8. Set the Enter Program Name field to the full path of the program to run After Distribution. In our case it is:

   ```
   /courier/images/aix/INed_After.sh
   ```

 See 5.4.1.5, “Create Program Options” on page 132 for a detailed description of this script.

9. When you select the **Upon Removal** push button, the following dialog is displayed:

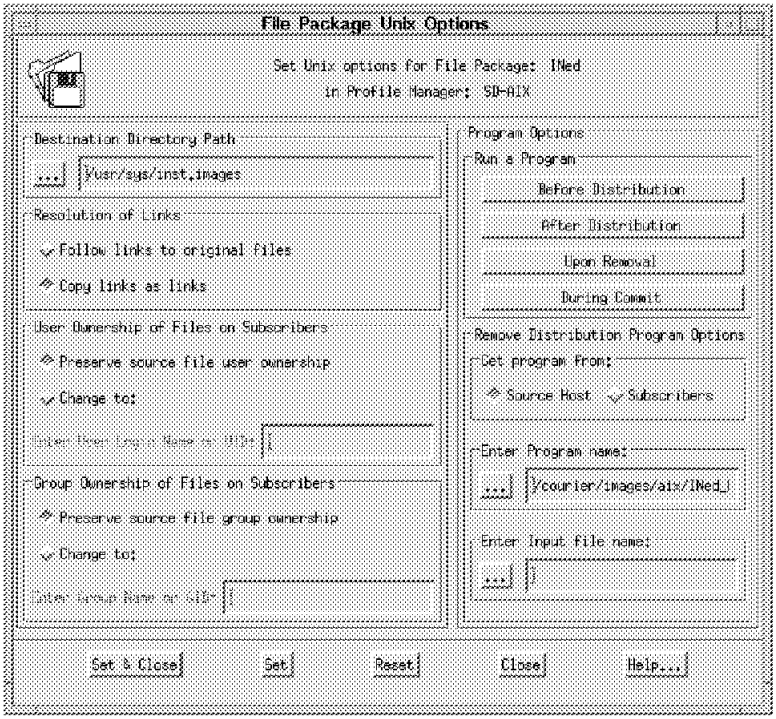


Figure 98. File Package UNIX Options (INed Remove)

10. Set the Get Program from field to specify the location of the program or procedure.

 Select the **Source Host** radio button to instruct Courier to copy the Upon Removal program from the source host to a temporary file on each subscriber. It will then run the programs on each subscriber to perform the removal. Next it removes each temporary file from each subscriber upon completion.

11. Set the Enter Program Name field to the full path of the program to run upon removal. In our case it is:

    ```
    /courier/images/aix/INed_Remove.sh
    ```

 See 5.4.1.5, “Create Program Options” on page 132 for a detailed description of this script.

12. When you select the **During Commit** push button, the following dialog is displayed:

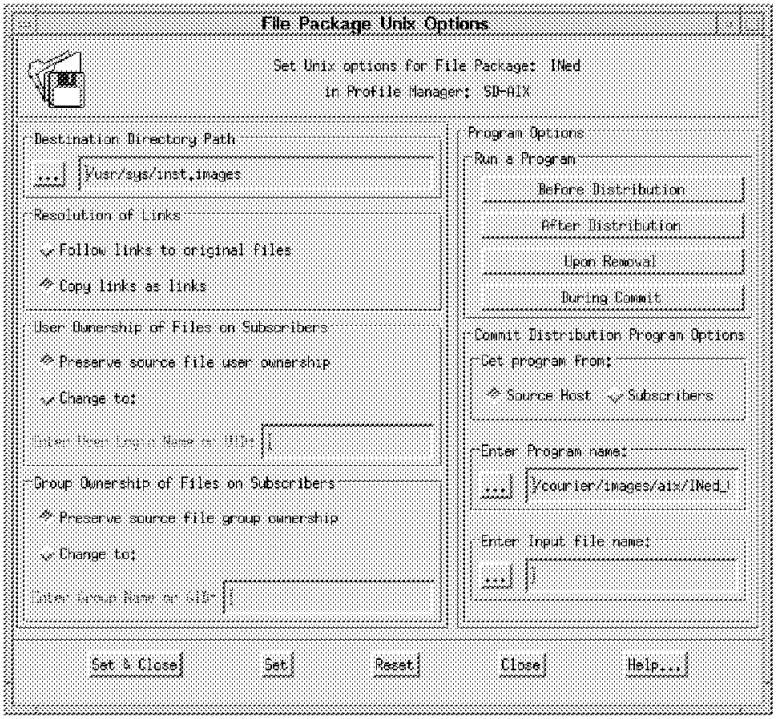


Figure 99. File Package UNIX Options (INed Commit)

13. Set the Get Program from field to specify the location of the program or procedure.

 Select the **Source Host** radio button to instruct Courier to copy the During Commit program from the source host to a temporary file on each subscriber. It will then run the programs on each subscriber to perform the commit. Next it removes each temporary file from each subscriber upon completion.

14. Set the Enter Program Name field to the full path of the program to run during commit. In our case it is:

    ```
    /courier/images/aix/INed_Commit.sh
    ```

 See 5.4.1.5, “Create Program Options” on page 132 for a detailed description of this script.

 Programs existence

 Courier does not check for the existence of any of those programs on either the source host or subscribers until software distribution time. If a program is not found at that time, then an error will be logged.

15. Select the **Set & Close** button to apply all of the changes and close the File Package UNIX Options windows.

16. Select the **Save & Close** button from the File Package Properties window to apply all of the changes to the file package.

5.4.1.4 Export/Import the File Package

We wanted to append to our log file /courier/logs/aix/INed.log on rs600011 all of the notification related to the distribution, commit or removal operations performed on the INed file package. Since the default for the notification is to *Replace the log file*, we would lose any previous log information.

To achieve this, we must use the export/import facility and save the file package properties in a text file known as the file package definition. The properties are represented as keywords and lists. Once you export a file package, you can set and modify the value of those keywords that are sometimes not available using the dialog. Therefore, we need to modify the value of the keyword append_log in the file package definition from n to y. You can then import the revised file package definition into the file package. Refer to 5.1.5, “Import/Export File Packages” on page 106 for more information.

1. From the SoftDist policy region double-click on the **SD-AIX** profile manager previously created in 5.3.3.1, “Create Profile Managers to Group Nodes and File Packages” on page 113.

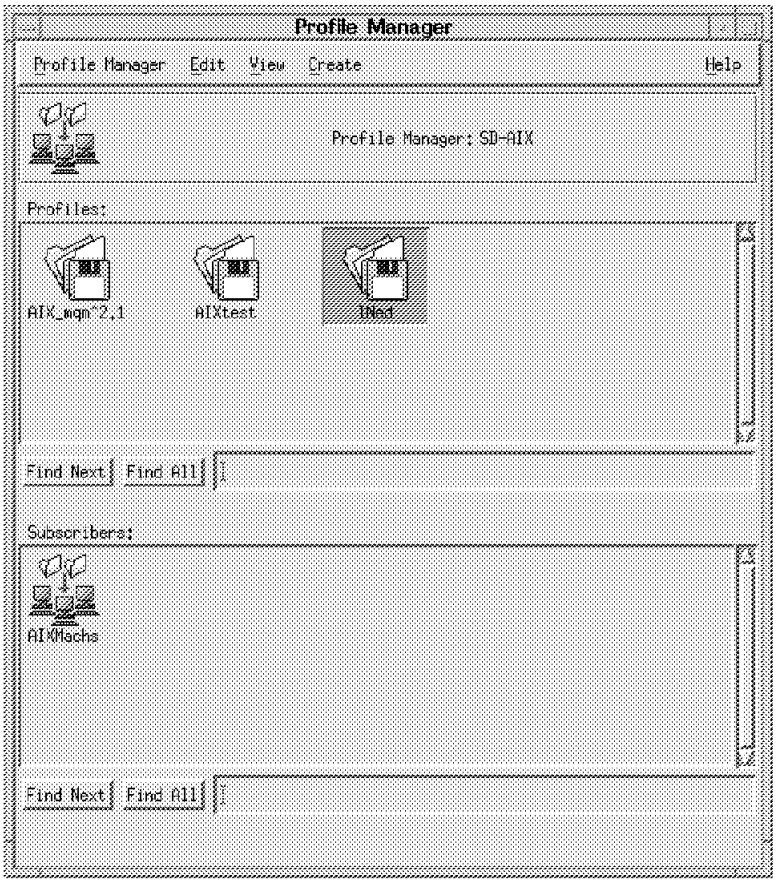


Figure 100. Profile Manager (SD-AIX)

2. Double-click on the **INed** file package. The TME displays the following dialog:

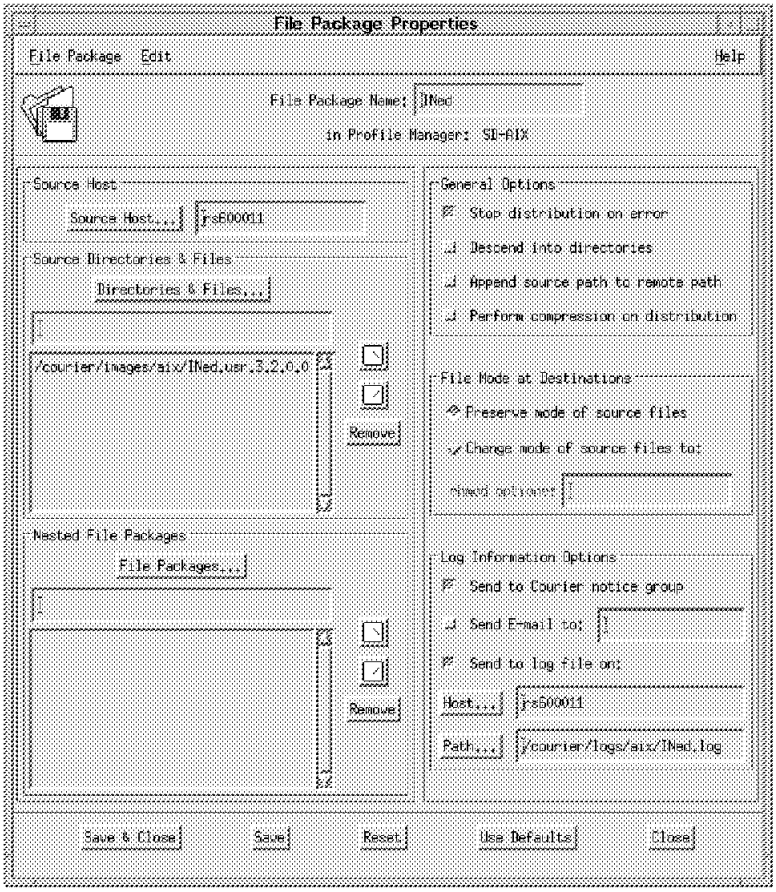


Figure 101. File Package Properties (INed)

3. Select **File Package = > Export...** to display the following panel:

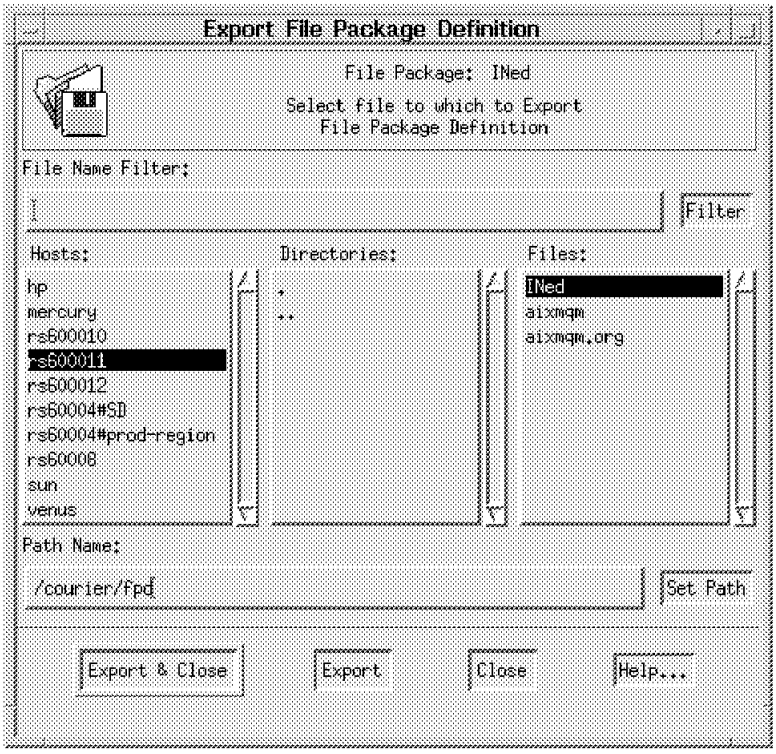


Figure 102. Export File Package Definition

4. This dialog enables you to save the file package definition to a text file.

 Set the Hosts field to rs600011 and the Path Name field to the file /courier/fpd/INed.

5. Select **Export & Close** to save the file package definition and to close the dialog.

6. Using an editor, open the /courier/fpd/INed file and modify the keyword append_log=n to append_log=y:

```
#*TFP-v2.02     Tivoli Filepack (version v2.02)
postproc=
preproc=
do_checksum=
do_compress=
modifiers=0
default_dest=
default_mtime=
rm_empty_dirs=n
stop_on_error=y
descend_dirs=n
keep_paths=n
rm_extraneous=n
follow_links=n
create_dirs=y
skip_older_src=n
no_overwrite=n
backup_fmt=
list_path=
nested_first=n
src_relpath=
file_cksums=n
unix_platform_prefix=/usr/sys/inst.images
nw_platform_prefix=
dos_platform_prefix=
win_platform_prefix=
post_notice=y
mail_id=
log_host=rs600011
log_file=/courier/logs/aix/INed.log
append_log=y
```

7. From Figure 101 on page 129, select **File Package = > Import...** to display the following panel:

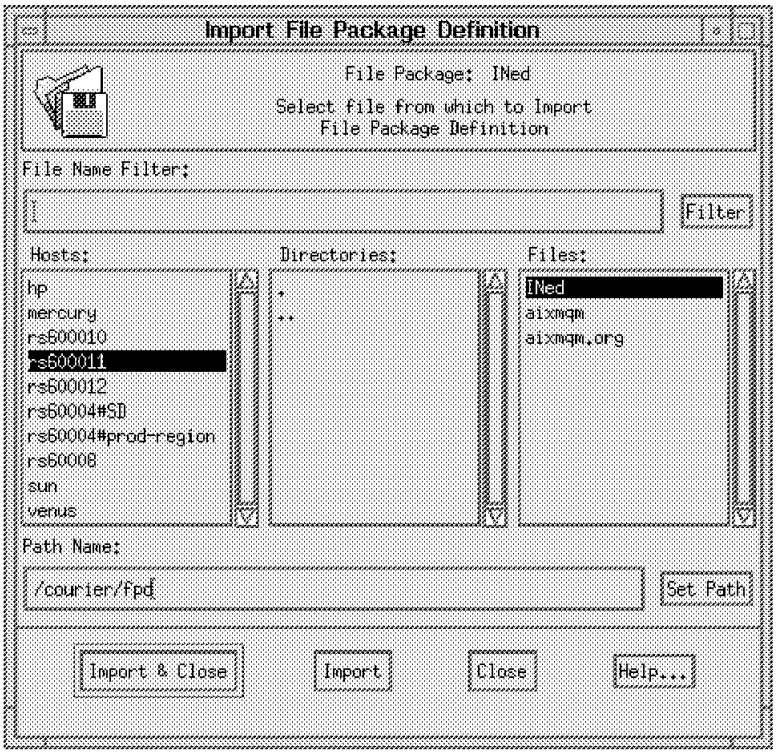


Figure 103. Import File Package Definition

8. This dialog enables you to import the file package definition into the file package.

 Set the Hosts field to rs600011 and the Path Name field to the file /courier/fpd/INed.

9. Select **Import & Close** to immediately import the file package definition and to close the dialog.

5.4.1.5 Create Program Options

After having defined the file package properties, it is necessary to create the program options. See 5.1.4, “Configuration Programs” on page 105 for more information on program options.

Those scripts will be stored in the source host rs600011 under the directory:

```
/courier/images/aix
```

We need to invoke the AIX command installp in order to install, remove and commit the INed product. Therefore, we wrote three scripts:

1. INed_After.sh - To Install the INed package
2. INed_Remove.sh - To Remove the INed package
3. INed_Commit.sh - To Commit the INed package

Note: We intend to distribute the INed file package to an AIX 3.2.5 system. Therefore, the scripts were written according to the behavior of installp in an AIX 3.2.5, which is different from an AIX 4.1 system.

The following are the listings of the scripts:

```
#!/bin/ksh
/etc/installp -qaFXd/usr/sys/inst.images INed 3.2.0.0 > /tmp/INed.log 2>&1
exit 0
```

Figure 104. The INed_After.sh script

```
#!/bin/ksh
/etc/installp -u INed 3.2.0.0 > /tmp/INed.log 2>&1
exit 0
```

Figure 105. The INed_Remove.sh script

```
#!/bin/ksh
/etc/installp -cX INed 3.2.0.0 > /tmp/INed.log 2>&1
exit 0
```

Figure 106. The INed_Commit.sh Script

5.4.2 Software Distribution of INed

You can now distribute the INed file package to its subscribers. This chapter describes how to:

- Distribute INed from rs600011 to target rs600012
- Remove INed from target rs600012
- Commit INed on target rs600012

As previously described in 5.3.2.1, “Configuring Repeaters” on page 111, rs600012 is a managed node defined as a repeater and it is managed by rs60008. Both of those nodes belong to TMR SD.

We will start the Distribution from the TMR Prod whose server is rs600011. Therefore, the INed File Package will be distributed through the path shown in the following picture:

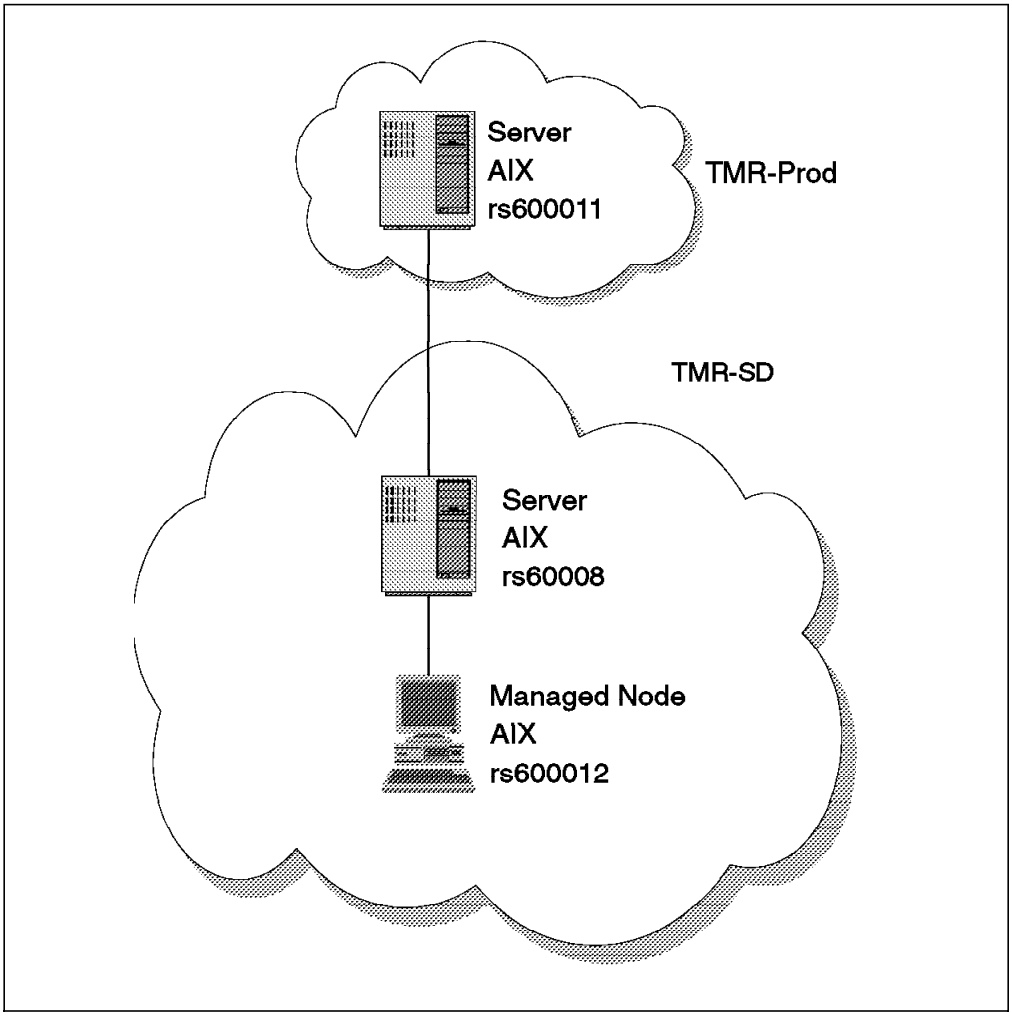


Figure 107. Distribution Path for the INed File Package

5.4.2.1 Distribute the File Package for INed

To distribute a file package you can use *drag and drop*, dragging the file package icon and dropping it into a managed node or a profile manager, the command line or the desktop.

We show a scenario using the desktop:

1. From the SoftDist policy pegion double-click on the **SD-AIX** profile manager.

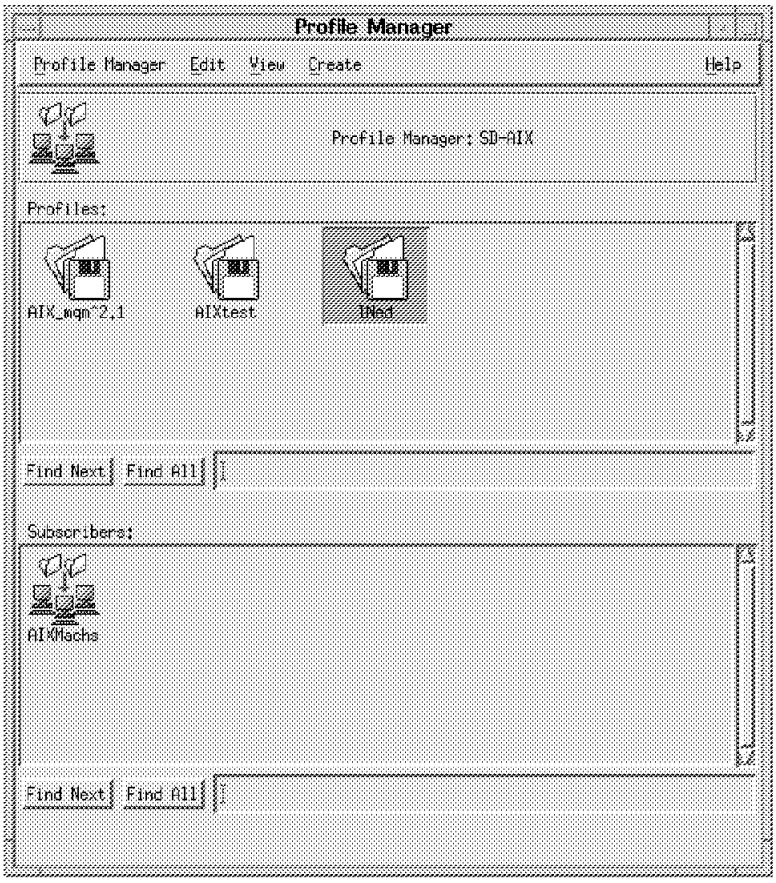


Figure 108. Profile Manager (SD-AIX)

2. Double-click on the **INed** file package icon to display the windows:

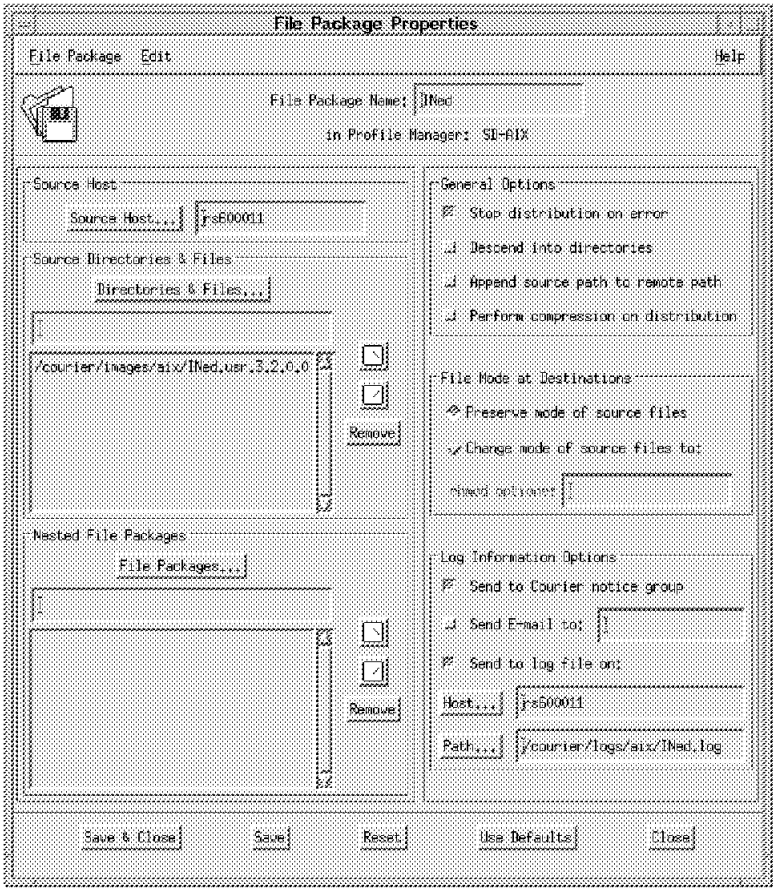


Figure 109. File Package Properties (INed)

3. Select **File Package = > Distribute...** to display the following panel:

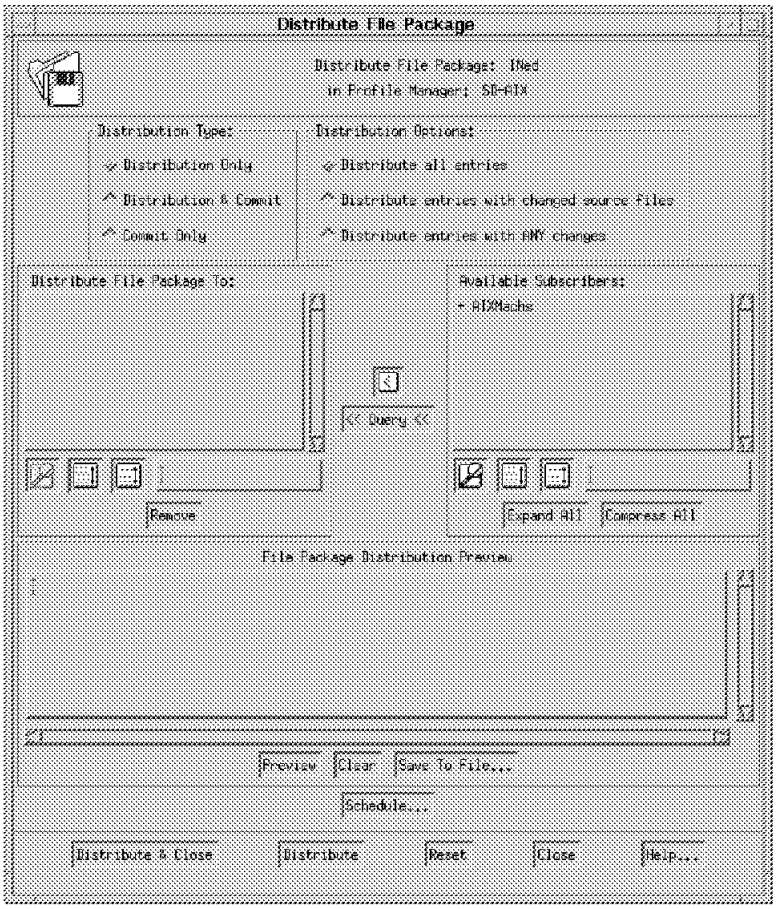


Figure 110. Distribute File Package (INed)

4. Set the Distribution Type field from the available list.

 Select **Distribution Only** to distribute only the file package. Any specified commit programs are not run during this operation.

5. Set the Distribution Options field to control which files in the file package are distributed.

 Select **Distribute all entries** to distribute all files and directories in the file package.

6. Fill the Distribute File Package To scrolling list with the subscribers to which you want the file package distributed.

 You can choose the subscribers from the Available Subscribers list and move them to the Distribute File Package To list using the arrow button.

 In our case AIXMachs is the profile manager in the Available Subscribers list. Therefore, we double-click on the profile manager, which is prefixed by the character **+** to display all of the managed nodes that subscribe to the profile manager. We then select **rs600012**.

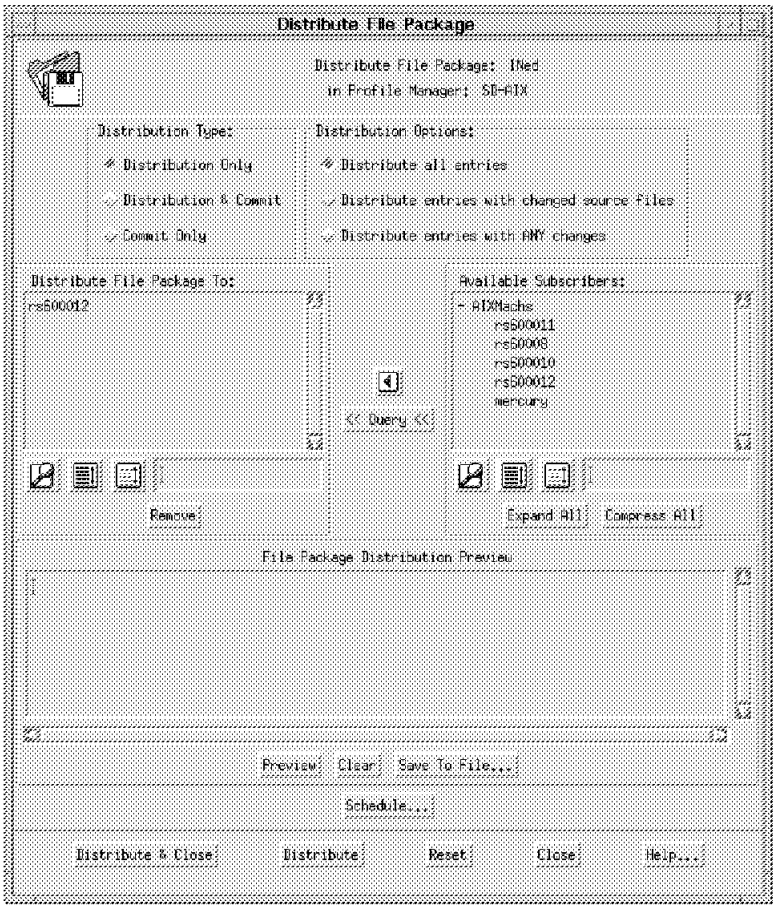


Figure 111. Distribute File Package (INed with Subscriber)

7. Select **Distribute & Close** to begin distributing the file package to target rs600012.

 Note: The dialog will not be dismissed until the distribution is complete. If the distribution fails for any of the subscribers, a pop-up dialog is displayed to inform you which subscribers failed distribution.

 Now Courier will send from the source to the target any source directories and files specified. Then it will start the After Distribution script specified in the file package.

Once the distribution is complete, you can check the distribution result by looking at the primary desktop panel, which will list in the Operation Status the software distribution activities:

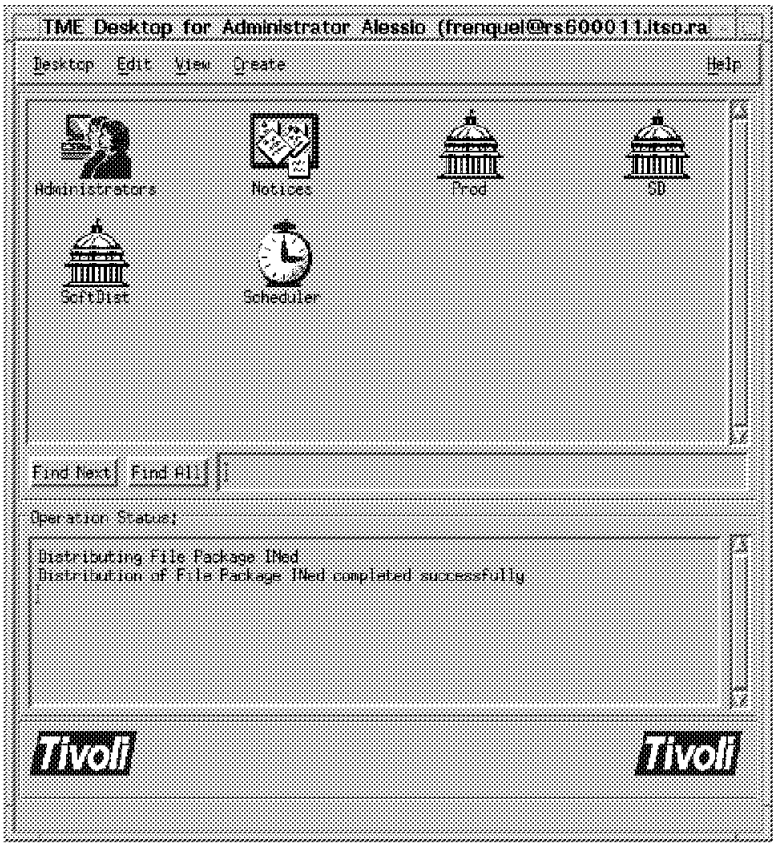


Figure 112. Desktop with Operation Status Messages

You also need to check the log file that we specified in the Log Information Options when we created the file package properties in 5.4.1.2, “Create the File Package Properties” on page 120.

The file /courier/logs/aix/INed.log on rs600011 will contain the following messages:

```
================
File Package:  ”INed”
Operation:     install  (m=5)
Finished:      Tue Aug 13 17:53:02 1996
--------
Source messages:
<none>
--------
rs600012: SUCCESS
temp script: unix_after: /tmp/unix_afterG5Lc5RCAAA
temp script: unix_commit: /tmp/unix_commitG5Lc5RCAAB
temp script: unix_removal: /tmp/unix_removalG5Lc5RCAAC
starting script: /tmp/unix_afterG5Lc5RCAAA
script complete: status=0
```

We have also specified in the File Package Properties dialog the Send to Courier notice group option, so that the Courier Notice group will include notice messages for file package operations.

1. From the primary desktop select the **Notices** icon to display the Read Notices menu.



Figure 113. Read Notices

2. Select the **Courier** option as shown in Figure 113. Then select the **Notice** message as shown in Figure 114.



Figure 114. Notice Group Messages (Courier)

Also the INed_After.sh script redirected the installp output to file /tmp/INed.log on the target machine. Therefore, you can check how the installp went by looking at the file on the target machine (in our case, rs600012).

```
installp:  Applying software for the "usr" part of the product
   INed 3.2.0.0.

installp Summary
--------------------
Name                      Fix Id   Part    Event      Result    State
-------------------------------------------------------------------------
INed.obj                           USR     APPLY      SUCCESS   APPLIED
```

5.4.2.2 Remove the File Package for INed

After a file package has been distributed on a target, it can be removed using the Remove function. The remove operation will remove any files that were previously distributed and also trigger any Before or After distribution program.

Note that Courier does not perform any check if the package was previously distributed on the target since it does not have any *history* associated with the file package.

1. From the SoftDist policy region double-click on the **SD-AIX** profile manager.

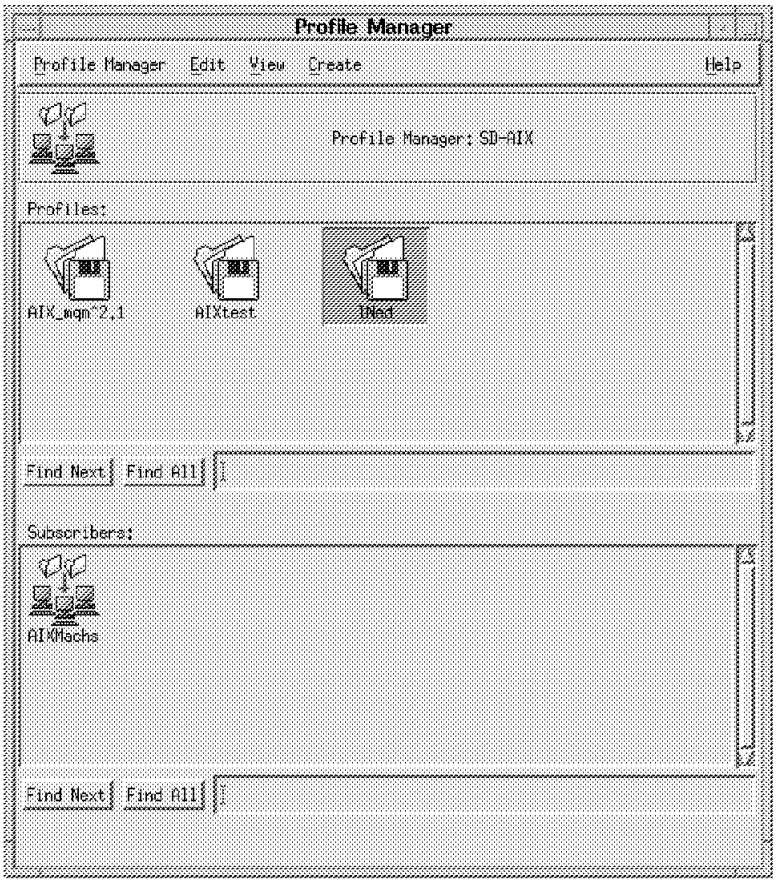


Figure 115. Profile Manager (SD-AIX)

2. Double-click on the **INed** file package icon to display the panel shown in Figure 115.

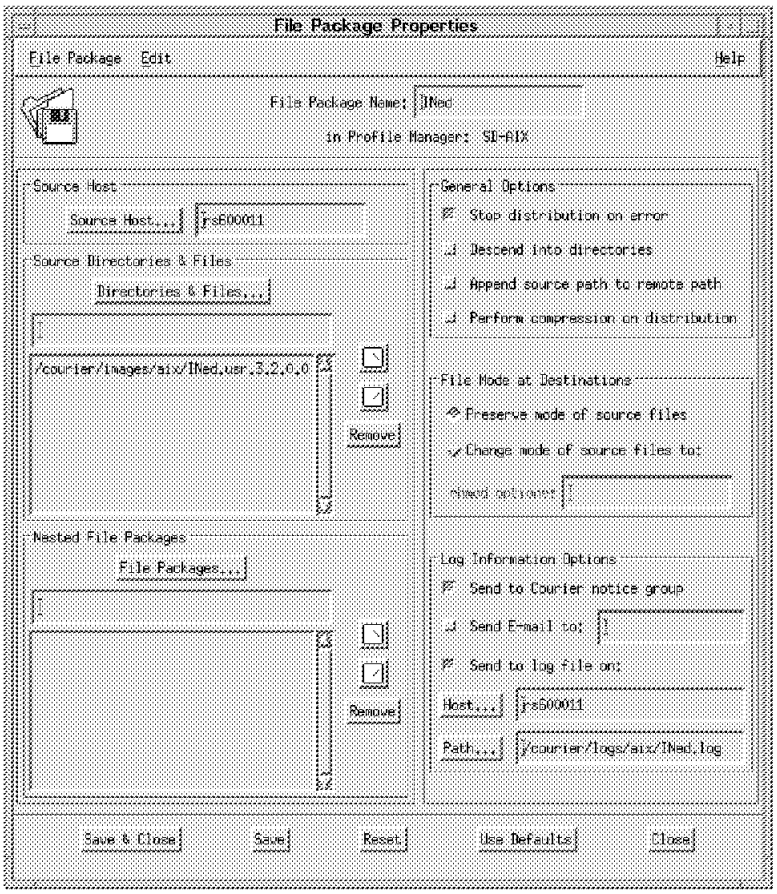


Figure 116. File Package Properties (INed)

3. Select **File Package = > Remove from Hosts...** to display the following panel:

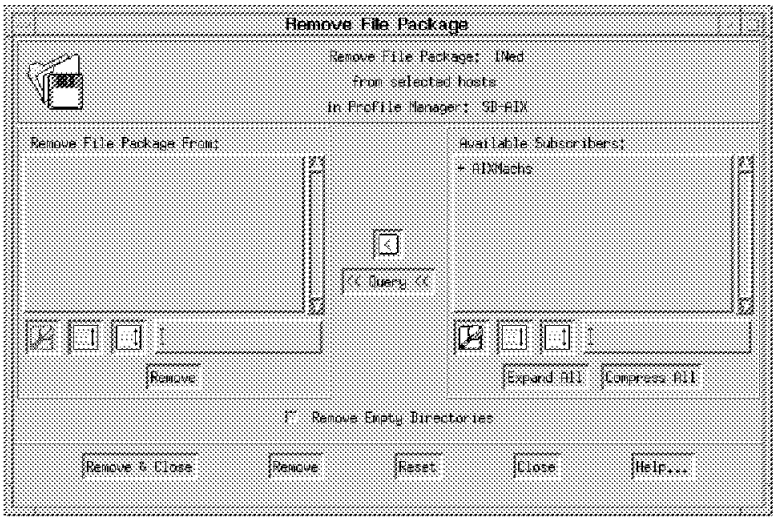


Figure 117. Remove File Package (INed)

4. Fill in the Remove File Package From scrolling list with the subscribers from which you want to remove the file package.

 You can choose the subscribers from the Available Subscribers list and move them to the Remove File Package From list using the arrow button.

 In our case we have AIXMachs as the profile manager in the Available Subscribers list. Therefore, we double-click on the profile manager, which is prefixed by the character + to display all of the managed nodes that subscribe to the profile manager. Then select **rs600012**:

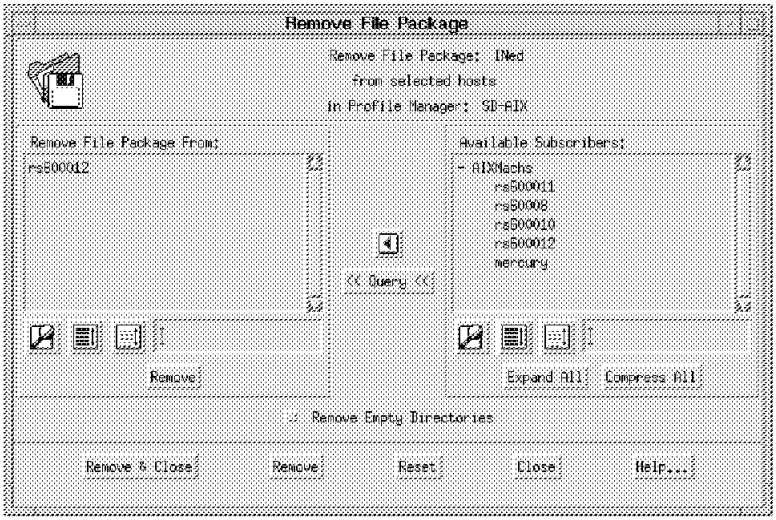


Figure 118. Remove File Package (INed from Subscriber)

5. Select **Remove & Close** to remove the file package from the subscriber rs600012.

 Note: The dialog will not be dismissed until the removal is complete. If the removal fails for any of the subscribers, a pop-up dialog is displayed to inform you on which subscribers the remove failed.

 Now Courier will start the Upon Removal script specified in the file package and then will remove from the target any source directories and files previously distributed.

 Remove Options

 Using the dialog you can only specify the Upon Removal script, which will be invoked only before the removal operation. If you want to run an after removal script, you need to export the file package, modify it, and then import it again. See 5.1.5, “Import/Export File Packages” on page 106 for more details.

Once the removal is complete, you can check the operation result by looking at the primary desktop panel, which will list the software distribution activities in the Operation Status field.

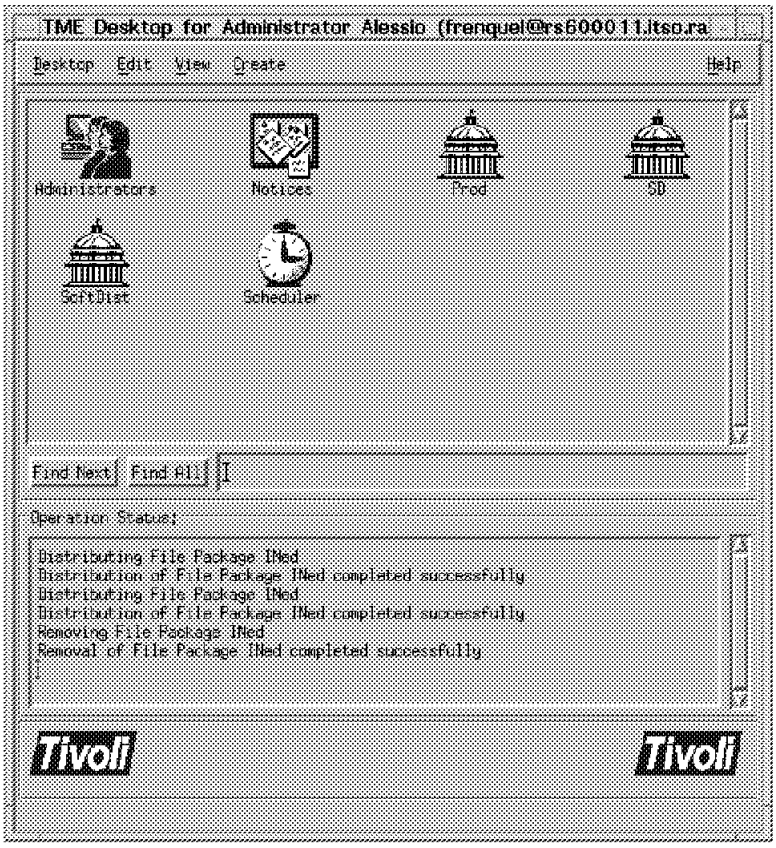


Figure 119. Desktop with Operation Status Messages

You also need to check the log file that we specified in the Log Information Options when we created the file package properties in 5.4.1.2, “Create the File Package Properties” on page 120.

The file /courier/logs/aix/INed.log on rs600011 will contain the following messages:

```
================
File Package:  "INed"
Operation:     uninstall (m=1)
Finished:      Tue Aug 13 18:04:25 1996
--------
Source messages:
<none>
--------
rs600012: SUCCESS
temp script: unix_after: /tmp/unix_afterG5Lc5RCAAA
temp script: unix_commit: /tmp/unix_commitG5Lc5RCAAB
temp script: unix_removal: /tmp/unix_removalG5Lc5RCAAC
starting script: /tmp/unix_removalG5Lc5RCAAC
script complete: status=0
```

We have also specified in the File Package Properties panel the Send to Courier notice group option, so that the Courier Notice group will include notice messages for file package operations.

The installation of this software is described in 5.4.2.1, “Distribute the File Package for INed” on page 134.

Also the INed_Remove.sh script redirected the installp output to the /tmp/INed.log file on the target machine. Therefore, you can check how the installp went by looking at the file on the target machine (in our case, rs600012).

```
installp:  Rejecting software for the "usr" part of the product
   INed.obj 3.2.0.0.

installp Summary
----------------
Name                    Fix Id   Part    Event       Result    State
-------------------------------------------------------------------------
INed.obj                         USR     REJECT      SUCCESS   AVAILABLE
```

5.4.2.3 Commit the File Package for INed

The Commit operation will only trigger the During Commit script contained in the file package in the specified target. It is important clarify that the Commit operation does nothing more than execute the Commit program. Note that Courier does not perform any check if the package was previously distributed on the target since it does not have any *history* associated with the file package.

1. From the SoftDist policy region double-click on the **SD-AIX** profile manager.

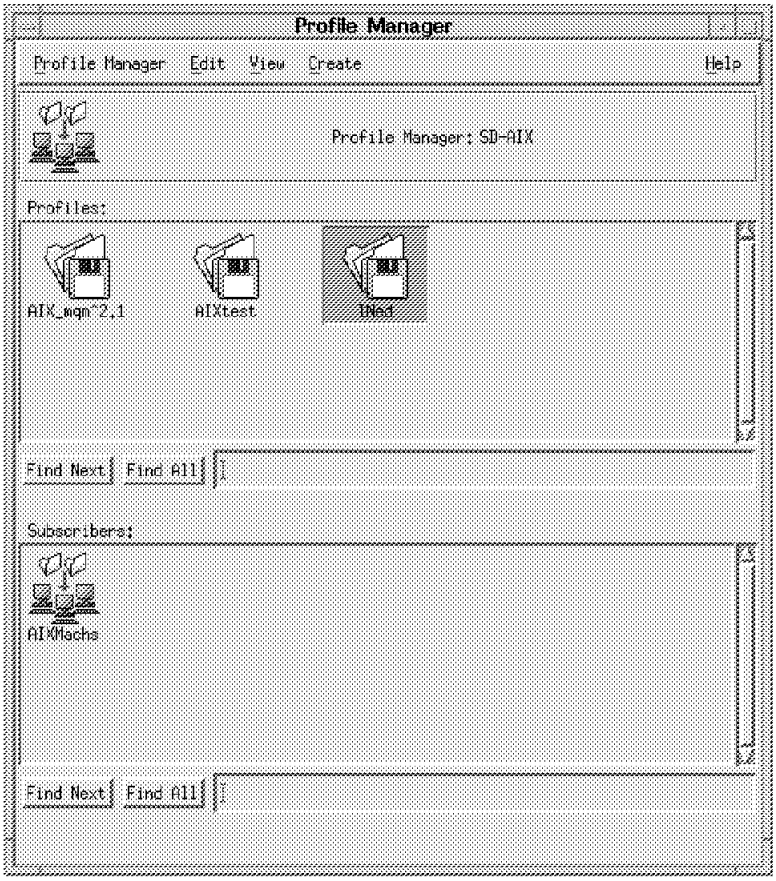


Figure 120. Profile Manager (SD-AIX)

2. Double-click on the **INed** file package icon to display the panel shown in Figure 121 on page 145.

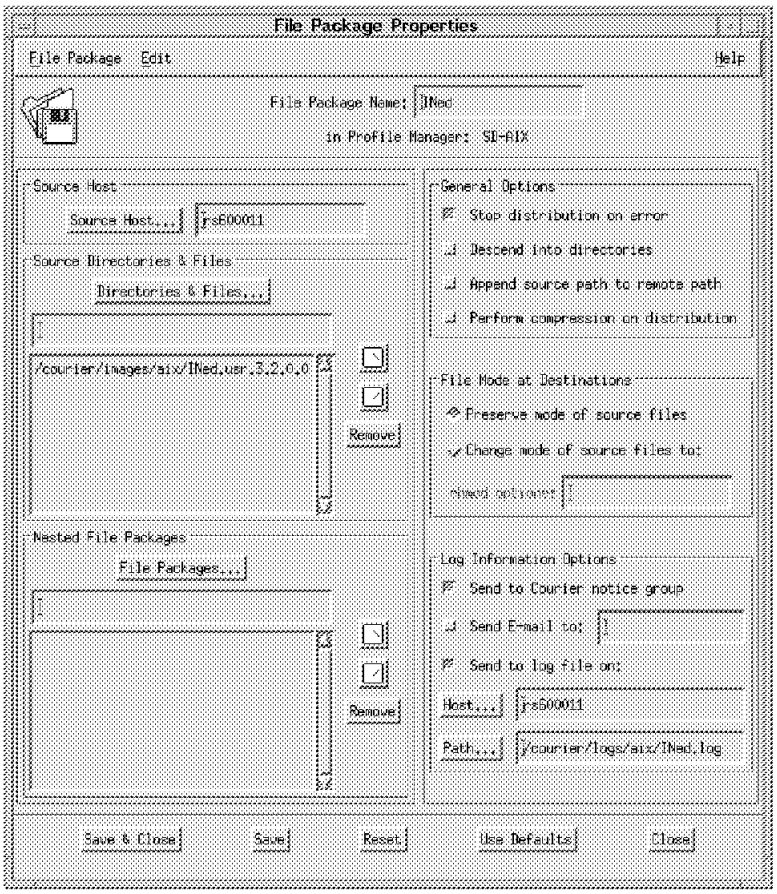


Figure 121. File Package Properties (INed)

3. Select **File Package = > Distribute...** to display the following panel:

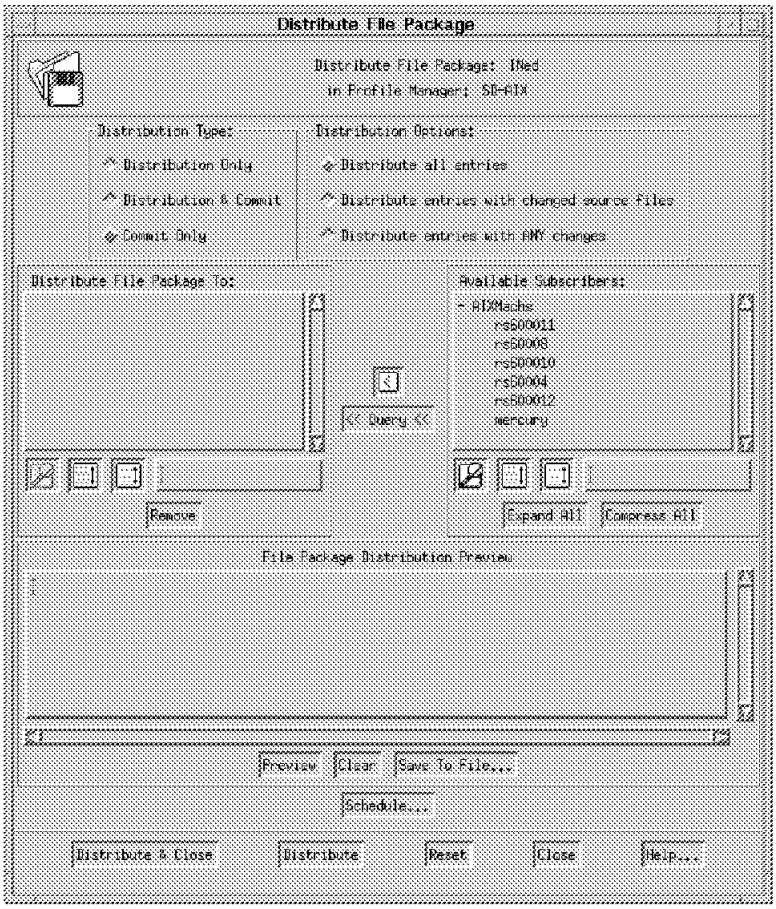


Figure 122. Distribute File Package (INed)

4. Set the Distribution Type field from the available list.

 Select the **Commit Only** radio button to run only the specified commit program on each subscriber.

5. Fill in the Distribute File Package To scrolling list with the subscribers on which you want to commit the file package.

 You can choose the subscribers from the Available Subscribers list and move them to the Distribute File Package To list using the arrow button.

 In our case, we have AIXMachs as the profile manager in the Available Subscribers list. Therefore, we double-click on the profile manager, which is prefixed by the character + to display all of the managed nodes that subscribe to the profile manager. We then select **rs600012**.

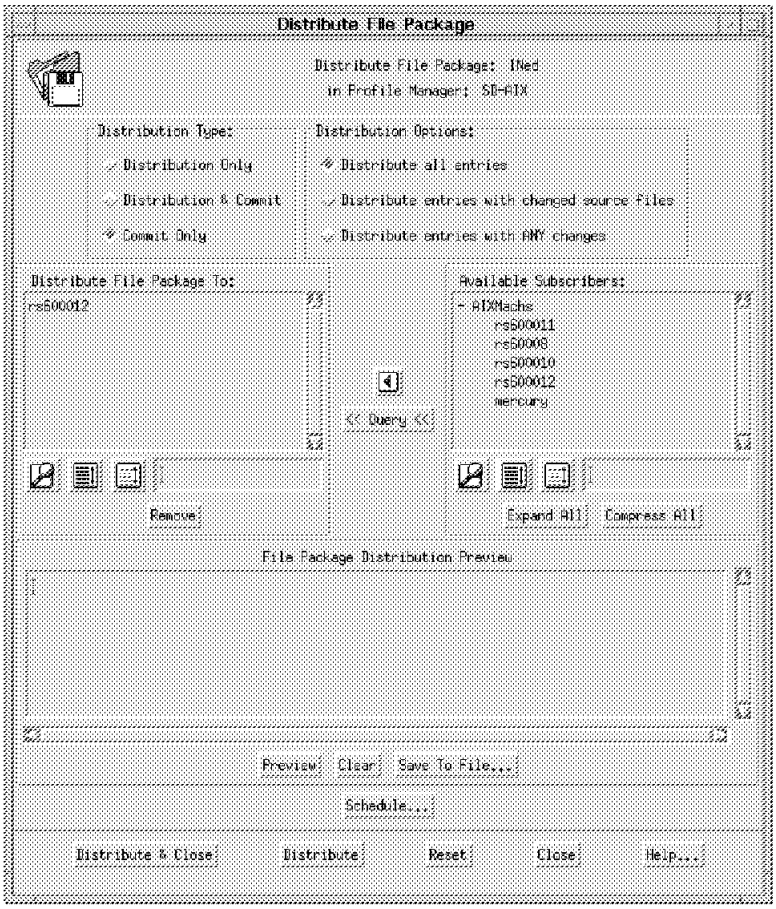


Figure 123. Commit File Package (INed with Subscriber)

6. Select **Distribute & Close** to commit the file package on the target rs600012.

 Note: The dialog will not be dismissed until the commit is complete. If the commit fails for any of the subscribers, a pop-up dialog is displayed to inform you which subscribers failed distribution.

 Now Courier will start the During Commit script specified in the file package.

Once the commit is complete, you can check the operation result by looking at the primary desktop panel, which will list the software distribution activities in the Operation Status field.

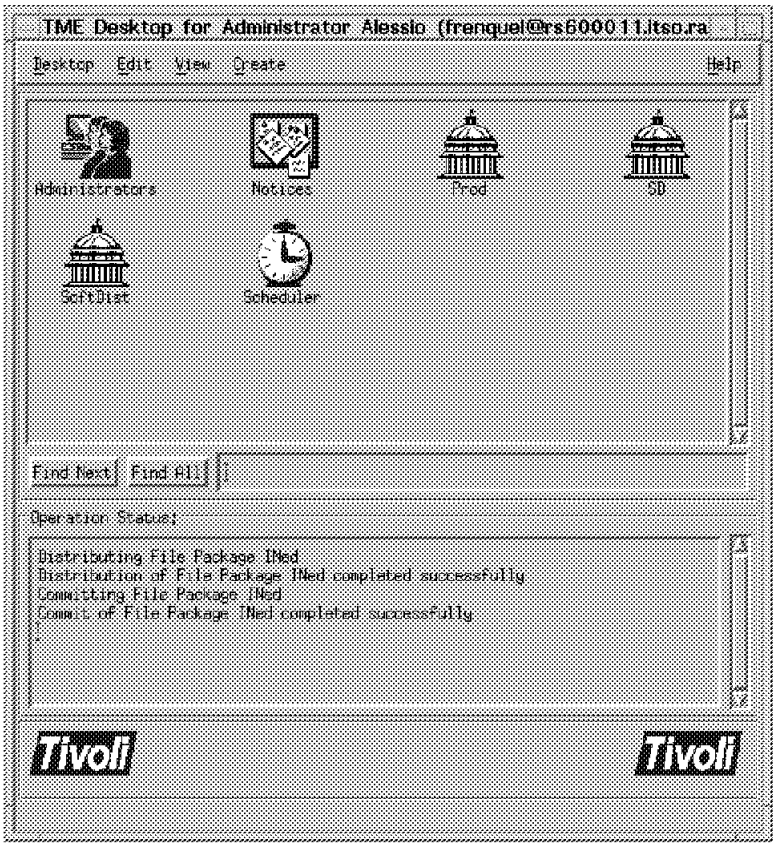


Figure 124. Desktop with Operation Status Messages

You also need to check the log file that we specified in the Log Information Options field when we created the file package properties in 5.4.1.2, “Create the File Package Properties” on page 120.

The file /courier/logs/aix/INed.log on rs600011 will contain the following messages:

```
================
File Package:  ”INed”
Operation:     install  (m=6)
Finished:      Tue Aug 13 18:43:25 1996
--------
Source messages:
<none>
--------
rs600012: SUCCESS
temp script: unix_after: /tmp/unix_afterG5Lc5RCAAA
temp script: unix_commit: /tmp/unix_commitG5Lc5RCAAB
temp script: unix_removal: /tmp/unix_removalG5Lc5RCAAC
starting script: /tmp/unix_commitG5Lc5RCAAB
script complete: status=0
```

We have also specified in the File Package Properties panel the Send to Courier notice group option, so that the Courier notice group will include notice messages for file package operations.

Check the Notices as previously described in 5.4.2.1, “Distribute the File Package for INed” on page 134.

Also the INed_Commit.sh script redirected the installp output to the /tmp/INed.log file on the target machine. Therefore, you can check how the installp went by looking at the file on the target machine (in our case, rs600012).

```
installp:  Committing software for the "usr" part of product
   INed 3.2.0.0.

installp Summary
----------------
Name                      Fix Id  Part    Event       Result      State
-------------------------------------------------------------------------
INed.obj                          USR     COMMIT      SUCCESS    COMMITTED
```

5.4.3 Create the MQSeries for AIX File Package

The MQSeries for AIX is an IBM product that provides application programming services that let application programs communicate with each other using *message queues*. For more information, refer to *MQSeries for AIX System Management Guide*, SC33-1373.

The MQSeries for AIX package is available in installp image format.

5.4.3.1 Planning the Installation

A good practice is to identify all of the activities and actions needed to install the product and possibly group them into:

- Before Distribution
- After Distribution
- Commit
- Remove

The following prerequisite steps are necessary before installing MQSeries for AIX. They will be coded in the Before Distribution script.

- Check that the machine has 80 MB free in the /usr filesystem.
- Check that the machine has 20 MB free in the /var filesystem.
- Create the *mqm* group.
- Create users *mqm* and *os2* and associate them to the group mqm.

First Failure Support Technology for AIX (referred to in this book as FFST/6000) is used to identify and analyze software events. FFST/6000 is an IBM licensed program that improves availabilty for IBM software applications by providing:

- Immediate software event notification
- First failure data capture for software events
- Automated event tracking and management

FFST events are unlike MQSeries events. FFST events occur when software probes embedded in the MQSeries product code are triggered as specific

conditions are met. These conditions are included in MQSeries for AIX code to assist with problem determination.

FFST is a corequisite to MQSeries for AIX installation. At installation time, MQSeries checks to determine whether or not FFST exists on your system and, if so, at what level.

FFST is supplied with MQSeries for AIX and is installed with MQSeries unless FFST/6000 is already installed at a current level on your system. For more information, refer to *MQSeries for AIX System Management Guide*, SC33-1373.

Therefore, it must also be installed on the AIX machine together with MQSeries for AIX. Once the installation of FFST/6000 and MQSeries is complete, other steps are necessary in order to initialize MQSeries for AIX. They will be coded in the After Distribution script.

- Invoke installp to install FFST/6000 and MQSeries.
- Create the message queue manager and define it as the default queue manager, using the command:

  ```
  crtmqm -q QMGRNAME
  ```

- Start the default queue manager, using the command:

  ```
  strmqm
  ```

- Initialize the queue manager invoking the script:

  ```
  runmqsc -e < /usr/lpp/mqm/samp/amqscoma.tst
  ```

- Append in /etc/services the following line:

  ```
  MQSeries     1414/tcp          #  for MQSeries
  ```

- Append in /etc/inetd.conf the following line:

  ```
  MQSeries  stream   tcp  nowait  mqm  /usr/lpp/mqm/bin/amqcrsta amqcrsta
  ```

- Refresh inetd to make the changes available, using the command:

  ```
  refresh -s inted
  ```

Before FFST/6000 and MQSeries are removed some additional steps are necessary. They will be coded in the Upon Removal script.

- Stop the queue manager, using the command:

  ```
  endmqm QMGRNAME
  ```

- Wait and then delete the queue manager, using the command:

  ```
  dltmqm QMGRNAME
  ```

- Invoke installp to remove FFST/6000 and MQSeries.

5.4.3.2 Create the Staging Area for the Source Files

The images needed to install the MQSeries for AIX are as follows:

- FFST/6000
- DynaText Browser
- MQSerier

Since the source host we used was rs600011, we copied the installp images into:

```
/courier/images/aix/mqm.obj
/courier/images/aix/epw121.bff
/courier/images/aix/dtextbrw.obj
```

5.4.3.3 Create the File Package Properties

1. From the SoftDist policy region double-click on the **SD-AIX** profile manager previously created in 5.3.3.1, “Create Profile Managers to Group Nodes and File Packages” on page 113.

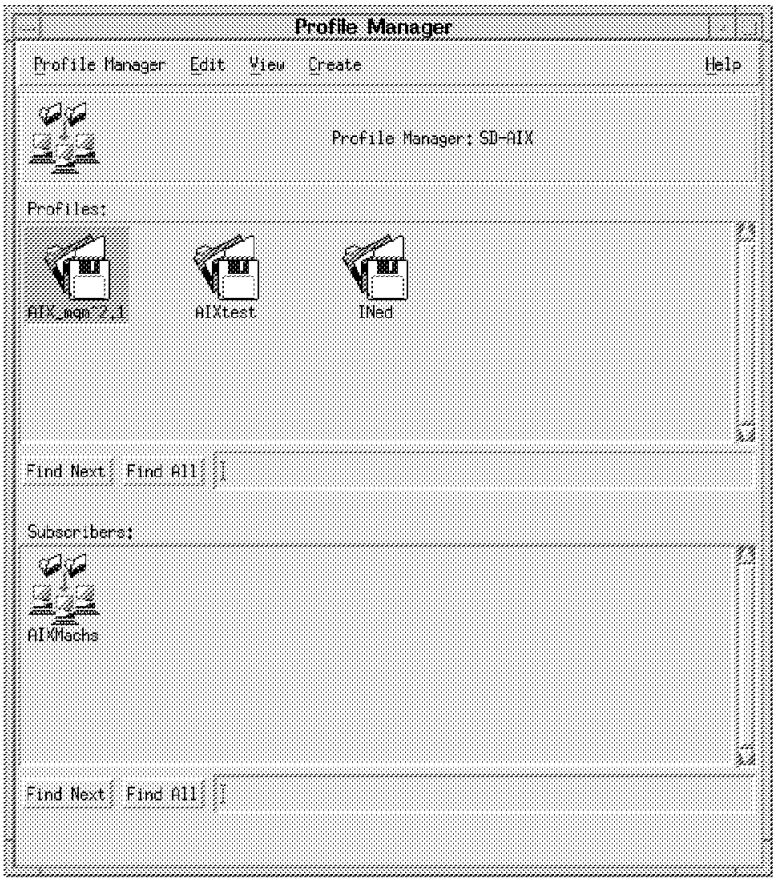


Figure 125. Profile Manager (SD-AIX)

2. Double-click on the **AIX_mqm^2.1** file package. The TME displays the following dialog:

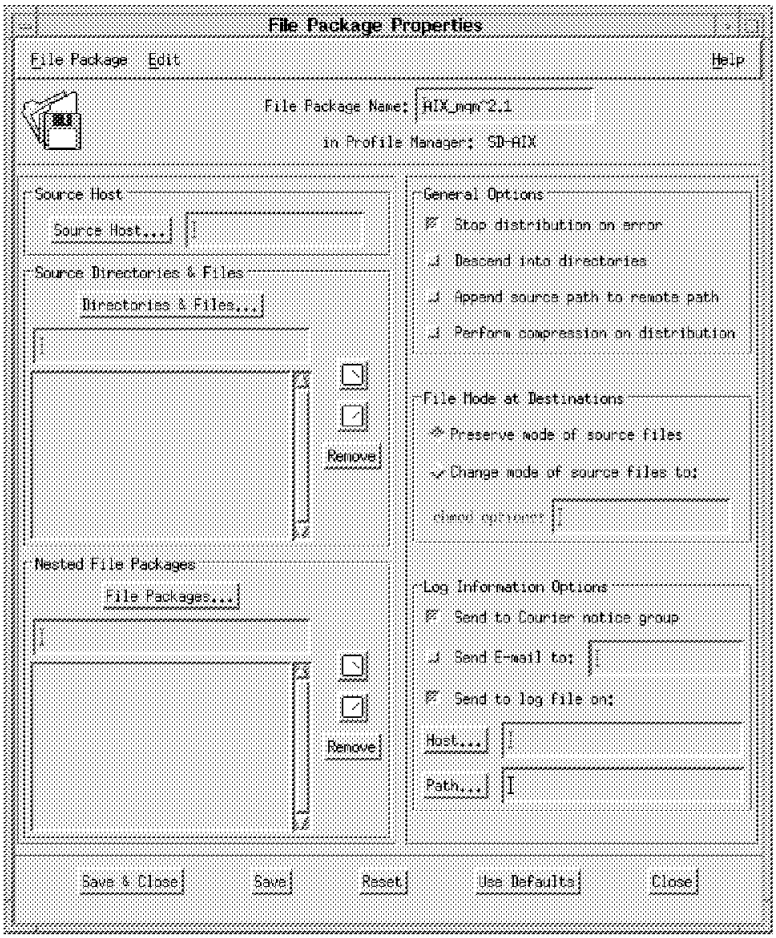


Figure 126. File Package Properties (MQSeries)

The File Package Properties panel shows the file package name, the directories and files, and the nested file packages to be included.

3. Set the Source Host field of the file package.

 Select the **Source Host...** push button to identify the name of the host on which the file package source files resides (for our project, rs600011).

 Source Host

 Only UNIX managed nodes are supported as source hosts.

4. Set the Source Directories & Files field of the file package.

 Select the **Directories & Files...** push button to identify the full path of the source files:

   ```
   /courier/images/aix/dtextbrw.obj
   /courier/images/aix/epw121.bff
   /courier/images/aix/mqm.obj
   ```

5. Set the Log Information Options field to control the logging activity.

 Select the **Send to Courier notice group** check box to have Courier post a notice, which includes an indication of success or failure of the operation for each target, to the Courier notice group when a file package operation occurs.

Select the **Send to log file on:** check box to have Courier place log information in the specified file when a file package operation occurs. An entry is made to the log each time a file package is distributed, committed or removed.

For our project, we defined the host to be rs600011 and the path to be /courier/logs/aix/mqm.log.

Log

You should select the **Send to log file on** check box. Otherwise, vital information regarding the distribution operations will never be logged.

6. We then left the defaults in all of the other boxes.

 The File Package Properties panel now looks as follows:

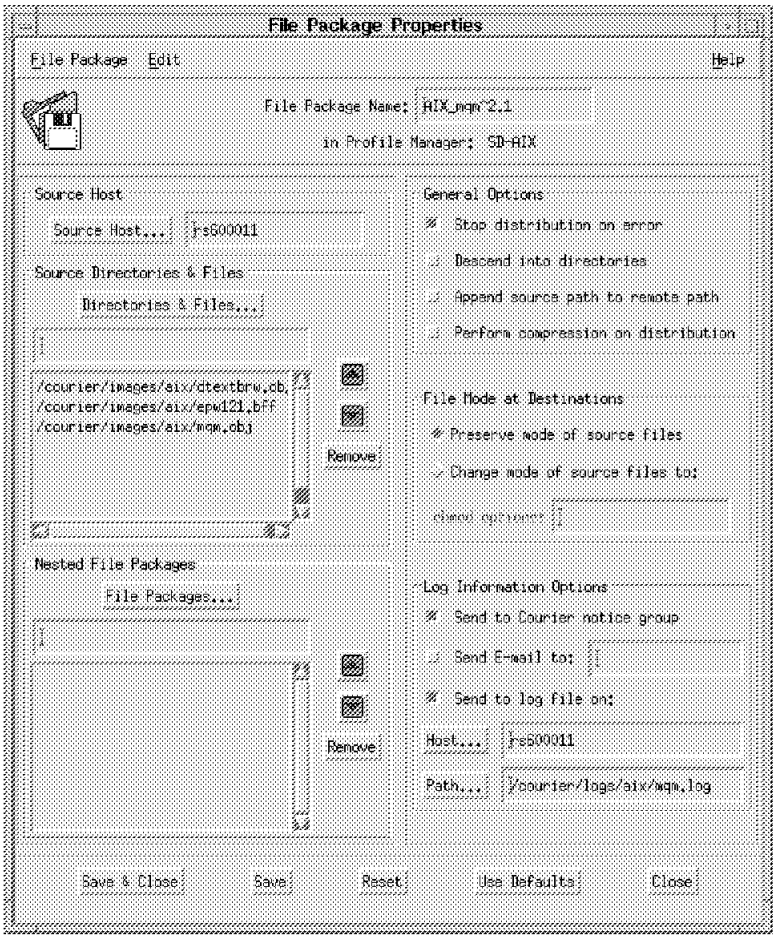


Figure 127. File Package Properties (MQSeries)

5.4.3.4 Create the File Package Platform-Specific Options

After defining the file package properties, define the platform-specific options of the file package. In our example we defined UNIX file package options.

1. From the File Package Properties panel shown in Figure 126 on page 152 select **Edit => Platform-Specific Options => UNIX Options...** to display the following panel:

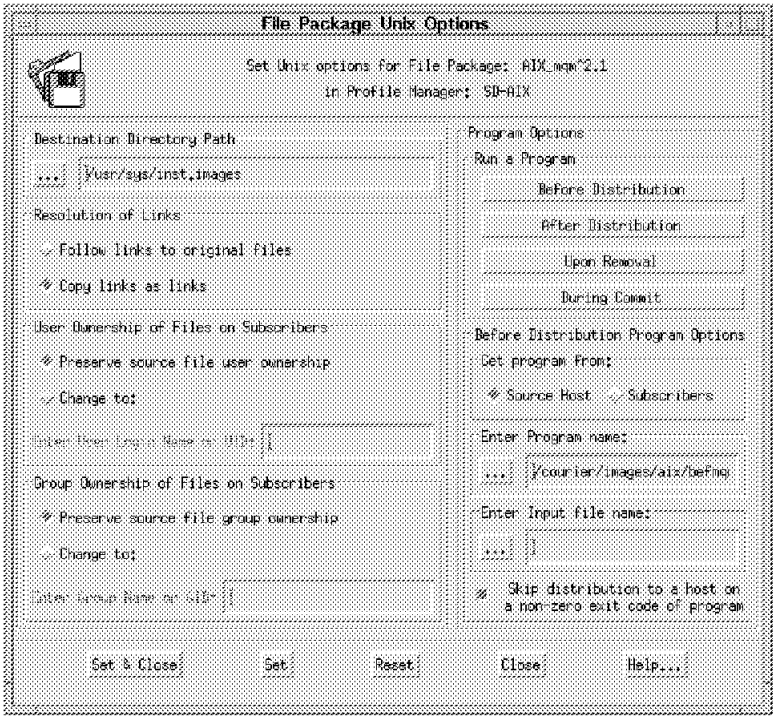


Figure 128. File Package UNIX Options (MQSeries Before)

2. Set the Destination Directory Path to the directory full path of the target to which the file package is distributed. In our case, we specified:

 `/usr/sys/inst.images`

3. Set the Resolution of Links to determine how symbolic links are handled during distribution and removal of file packages.

 Select the **Copy links as links** radio button to create symbolic links to the original files at the destination.

4. Set the Program Options using the Run a Program buttons.

 These buttons display options that enable you to run programs or procedures, such as, C programs, shell scripts, and Perl scripts, on subscribers before or after distributing, removing or

 The MQSeries package needs the following program to be run:

 Before Distribution This is a script to check and create the environment.

 After Distribution This is a script to install the package and run some commands.

 Upon Removal This is a script to run some commands and remove the package.

5. Now we need to set the Before Distribution Program options on Figure 128.

 Set the Get Program from field to specify the location of the program or procedure.

 Select the **Source Host** radio button to instruct Courier to copy the Before Distribution program from the source host to a temporary file on each subscriber. It will then run the programs on each subscriber after the distribution and remove each temporary file from each subscriber upon completion.

6. Set the Enter Program name field to the full path of the program to run Before Distribution. In our case it is:

   ```
   /courier/images/aix/befmqm.sh
   ```

 See 5.4.3.5, “Create Program Option” on page 157 for a detailed description of this script.

7. Select the **Skip distribution to a host on a non-zero exit code of program** check box, since we are running a Before script that checks for disk availability on the target machine.

 This will allow you to skip the distribution to the subscriber if the script fails and return a non-zero exit code. See 5.4.3.5, “Create Program Option” on page 157 for a detailed description of this script.

8. When you select the **After Distribution** button, the following dialog is displayed:

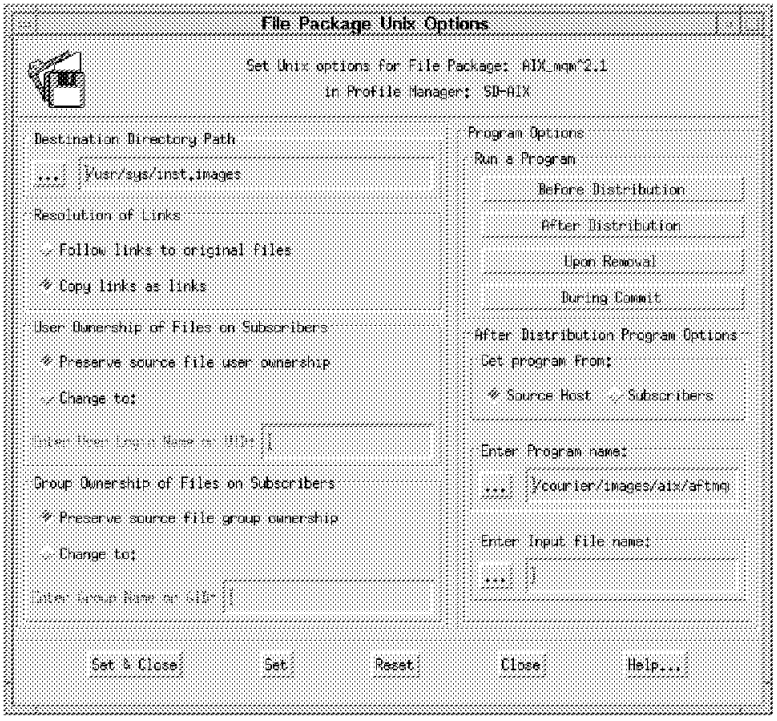


Figure 129. File Package UNIX Options (MQSeries After)

9. Set the Get Program from field to specify the location of the program or procedure.

 Select the **Source Host** radio button to instruct Courier to copy the After Distribution program from the source host to a temporary file on each subscriber. It will then run the programs on each subscriber after the distribution and remove each temporary file from each subscriber upon completion.

10. Set the Enter Program name field to the full path of the program to run After Distribution. In our case it is:

    ```
    /courier/images/aix/aftmqm.sh
    ```

 See 5.4.3.5, “Create Program Option” on page 157 for a detailed description of this script.

11. When you select the **Upon Removal** button, the following dialog is displayed:

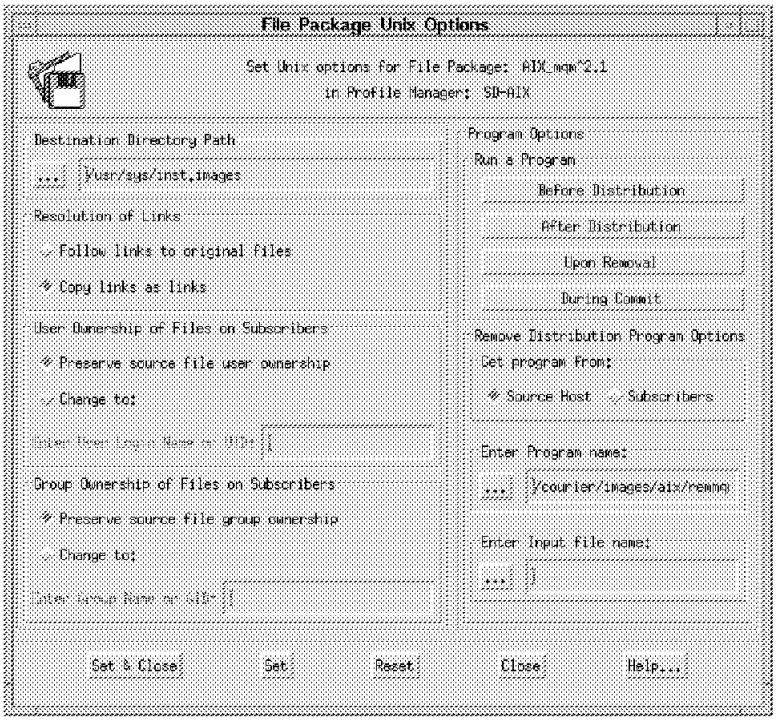


Figure 130. File Package UNIX Options (MQSeries Remove)

12. Set the Get Program from field to specify the location of the program or procedure.

 Select the **Source Host** radio button to instruct Courier to copy the Upon Removal program from the source host to a temporary file on each subscriber. It will then run the programs on each subscriber to perform the remove and remove each temporary file from each subscriber upon completion.

13. Set the Enter Program name field to the full path of the program to run upon removal. In our case it is:

    ```
    /courier/images/aix/remmqm.sh
    ```

 See 5.4.3.5, “Create Program Option” on page 157 for a detailed description of this script.

 Program′s existence

 Courier does not check for the existence of any of those programs on either the source host or subscribers until the time of software distribution. If a program is not found at that time, then an error will be logged.

14. Select **Set & Close** to apply all of the changes and close the File Package UNIX Options window.

15. Select **Save & Close** from the File Package Properties window to apply all of the changes to the file package.

5.4.3.5 Create Program Option

After having defined the file package properties, it is necessary to create the program options. See 5.4.3, “Create the MQSeries for AIX File Package” on page 149 for a complete description of the requirements for the program options.

Those scripts will be stored on the source host rs600011 under the directory:

```
/courier/images/aix
```

Now we need to check the disk space. Invoke the AIX utility `installp` in order to install and remove the MQSeries for AIX product and also run some commands. Therefore, we write three scripts:

1. befmqm.sh - to run some Pre-Install commands
2. aftmqm.sh - to install the MQSeries package
3. remmqm.sh - to remove the MQSeries package

Note: We intend to distribute the MQSeries file package to an AIX 4.1 system. Therefore, the scripts were written according to the behavior of installp in an AIX 4.1, which is different from an AIX 3.2.5 system. The differences are due to the design of the ODM database. When installing a new level of a product on an AIX 4.1 system, the status is always set to COMMIT in the ODM database. For this reason we do not show a commit scenario for the MQSeries. To remove the MQSeries it will then be necessary to invoke the UNINSTALL, which will remove the COMMITTED software.

Figure 131 gives a listing of the scripts.

```
#!/bin/ksh

#check that /usr has 80 Mb available and that /var has 20 Mb available

usr_needed=80000
var_needed=20000

usr_ava=`df -k /usr | awk '/\/usr/ {print($3)}'`
var_ava=`df -k /var | awk '/\/var/ {print($3)}'`

if [[ $usr_ava -le $usr_needed ||
 $var_ava -le $var_needed ]]
then
  echo "MQS94001 : $(hostname) Not enough space in /usr or /var file
systems. The installation is aborted." >> /tmp/mqm.log
  exit 1
fi

mkgroup '-A' mqm
mkuser groups='mqm' admgroups='mqm' mqm
mkuser groups='mqm' admgroups='mqm' os2
exit 0
```

Figure 131. The MQSeries befmqm.sh Script

```
#!/bin/ksh

cd /usr/sys/inst.images
/usr/sbin/inutoc .

/usr/sbin/installp -qagXd/usr/sys/inst.images dtext 2.3.0.1.all
epw 1.2.1.0.all mqm 2.2.1.0.all > /tmp/mqm.log 2>&1

if [[ $? -ne 0 ]]
then
  echo "MQS94002 : $(hostname) installp failed. Installation
aborted." >> /tmp/mqm.log
  exit 1
fi

crtmqm -q QMGRNAME >> /tmp/mqm.log 2>&1
if [[ $? -ne 0 ]]
then
  echo "MQS94003 : $(hostname) Create Queue Manager failed" >>
/tmp/mqm.log
  exit 1
fi

strmqm >> /tmp/mqm.log 2>&1
if [[ $? -ne 0 ]]
then
  echo "MQS94004 : $(hostname) Start Queue Manager failed" >>
/tmp/mqm.log
  exit 1
fi

runmqsc -e < /usr/lpp/mqm/samp/amqscoma.tst >> /tmp/mqm.log 2>&1

echo "MQSeries        1414/tcp          # for MQSeries" >> /etc/services

echo "MQSeries  stream  tcp    nowait   mqm  /usr/lpp/mqm/bin/amqcrsta
amqcrsta" >>/etc/inetd.conf

refresh -s inetd >> /tmp/mqm.log 2>&1
echo "MQS94000 : $(hostname) Installation completed successfully" >>
/tmp/mqm.log
exit 0
```

Figure 132. The MQSeries aftmqm.sh Script

```
#!/bin/ksh

endmqm QMGRNAME >> /tmp/mqm.log 2>&1

if [[ $? -ne 0 ]]
then
  echo "MQS94005 : $(hostname) End Queue Manager failed. Remove
aborted." >> /tmp/mqm.log
  exit 1
fi

echo "Delay 2 minutes for queue manager to end...." >> /tmp/mqm.log

sleep 120

dltmqm QMGRNAME  >> /tmp/mqm.log 2>&1
if [[ $? -ne 0 ]]
then
  echo "MQS94006 : $(hostname) Delete Queue Manager failed. Remove
aborted." >> /tmp/mqm.log
  exit 1
fi

rmuser -p os2
rmuser -p mqm
rmgroup mqm

/usr/sbin/installp -uX dtext.brwsr.obj 2.3.0.1 epw.adm 1.2.1.0 epw.bin 1.
2.1.0 epw.doc 1.2.1.0 epw.kext 1.2.1.0 epw.lib 1.2.1.0 mqm.De_DE 2.2.1.0
mqm.Es_ES 2.2.1.0 mqm.Fr_FR 2.2.1.0 mqm.Ja_JP 2.2.1.0 mqm.aix_client 2.2.
1.0 mqm.base 2.2.1.0 mqm.books 2.2.1.0 mqm.dt_client 2.2.1.0 mqm.dtext_bo
oks.2.1.0 mqm.runtime 2.2.1.0 mqm.samples 2.2.1.0 mqm.server 2.2.1.0 > /t
mp/mqm.log 2>&1

if [[ $? -ne 0 ]]
then
  echo "MQS95007 : $(hostname) installp failed during uninstall" >>
/tmp/mqm.log
  exit 1
fi

echo "MQS95008 : $(hostname) Remove completed successfully" >>
/tmp/mqm.log
exit 0
```

Figure 133. The MQSeries remmqm.sh Script

5.4.4 Software Distribution of MQSeries for AIX

You can now distribute the MQSeries for AIX file package to its subscribers. This chapter describes how to:

- Distribute MQSeries from rs600011 to target mercury
- Remove MQSeries from target mercury

As previously described in 5.3.2.1, “Configuring Repeaters” on page 111, mercury is a managed node. It is managed by rs600012, which is defined as a repeater. Both of those nodes belong to TMR SD.

We will start the distribution from the TMR Prod whose server is rs600011. Only one copy of the MQSeries file package will be distributed through the path shown in the following picture:

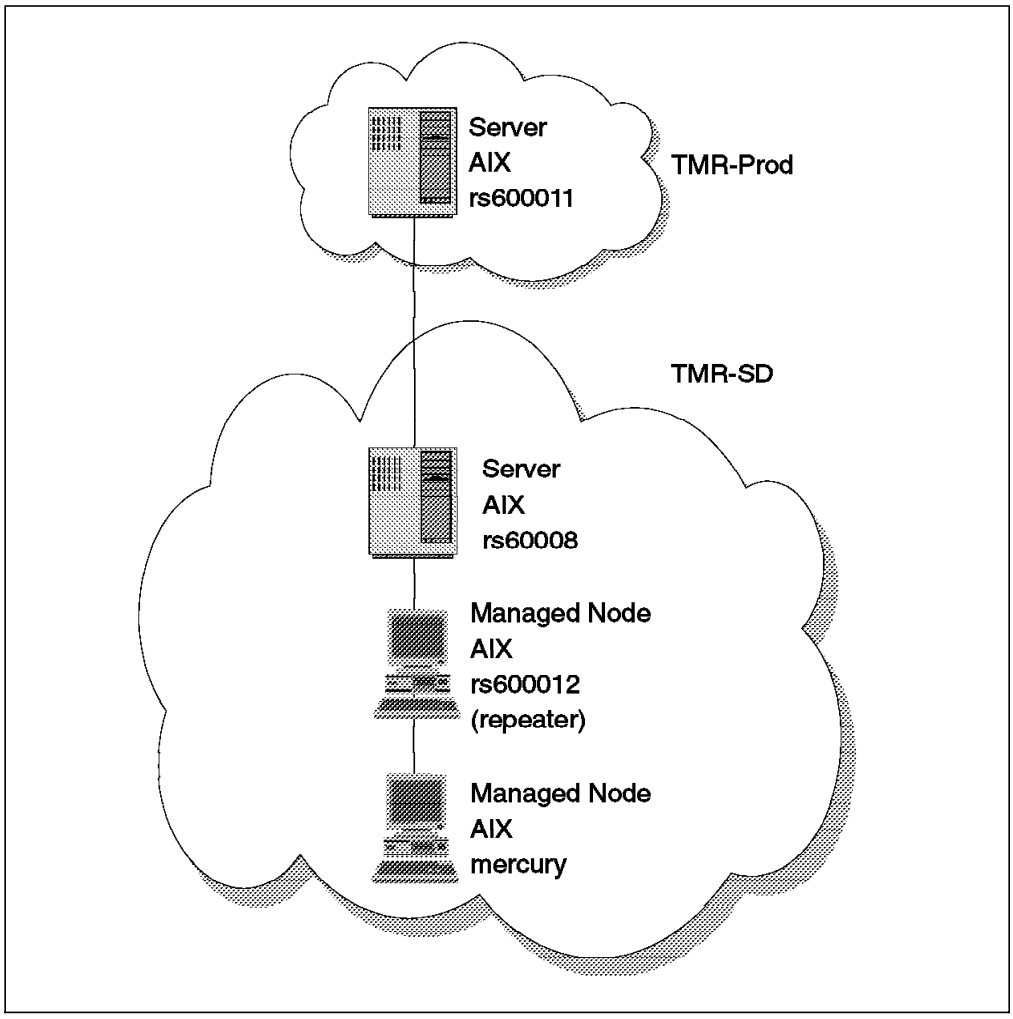


Figure 134. Distribution Path for the MQSeries for AIX File Package

5.4.4.1 Distribute the File Package for MQSeries

To distribute a file package you can use *drag and drop*, dragging the file package icon and dropping it into a managed node or a profile manager, the command line or the desktop.

We show a scenario using the desktop:

1. From the SoftDist policy region double-click on the **SD-AIX** profile manager.

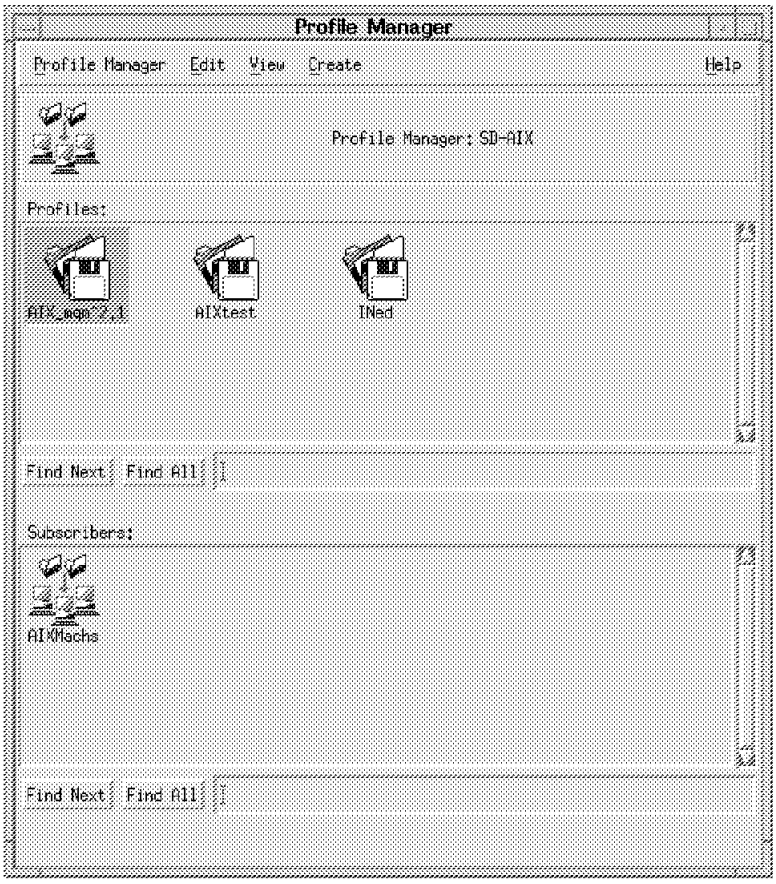


Figure 135. Profile Manager (SD-AIX)

2. Double-click on the **AIX_mqm^2.1** file package icon to display the windows:

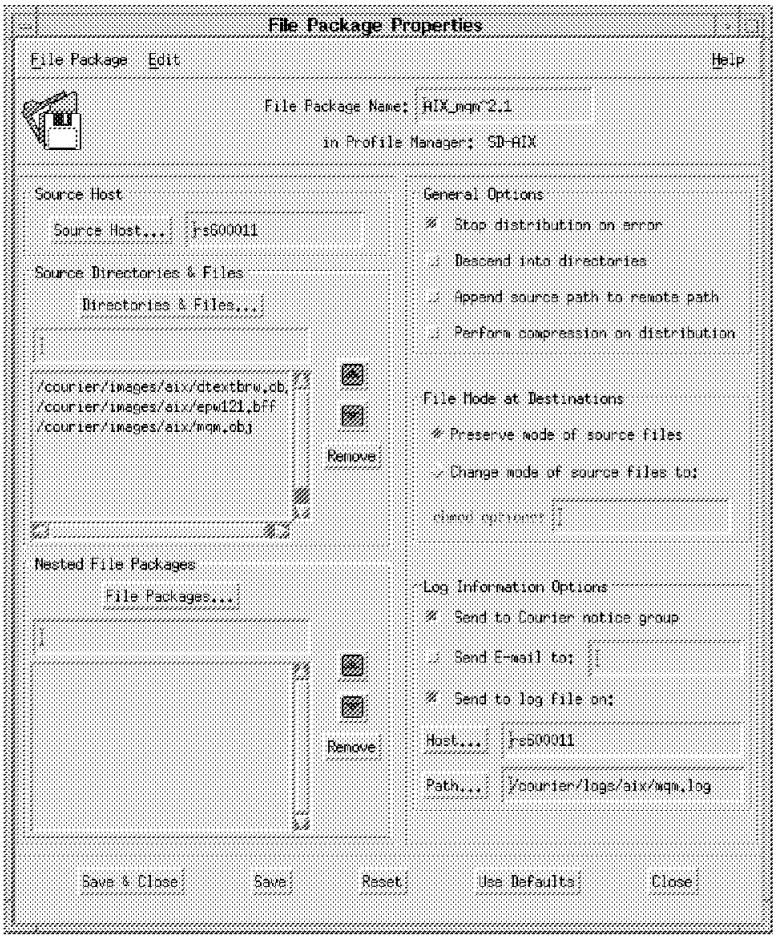


Figure 136. File Package Properties (MQSeries)

3. Select **File Package = > Distribute...** to display the following panel:

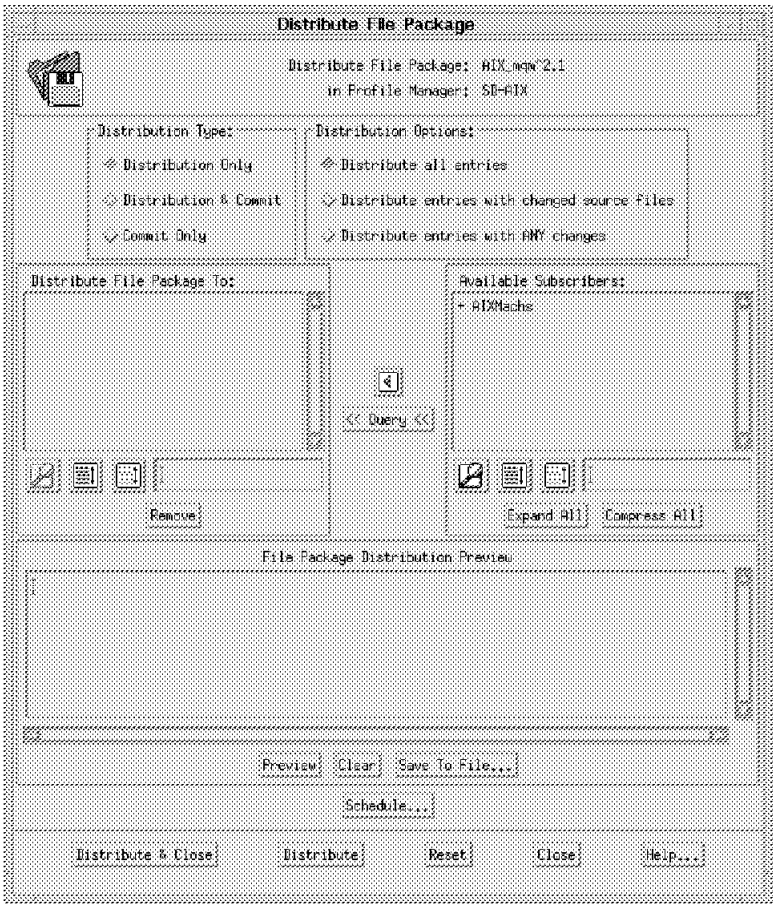


Figure 137. Distribute File Package (MQSeries)

4. Set the Distribution Type from the available list.

 Select **Distribution Only** to distribute only the file package. Any specified commit programs are not run during this operation.

5. Set the Distribution Options to control which files in the file package are distributed.

 Select **Distribute all entries** to distribute all files and directories in the file package.

6. Fill in the Distribute File Package To scrolling list with the subscribers to which you want to distribute the file package.

 You can choose the subscribers from the Available Subscribers list and move them to the Distribute File Package To list using the arrow button.

 In our case we have AIXMachs as the Profile Manager in the Available Subscribers list. Therefore, we double-click on the profile manager, which is prefixed by the character + to display all of the managed nodes that subscribe to the profile manager. We then select **mercury**.

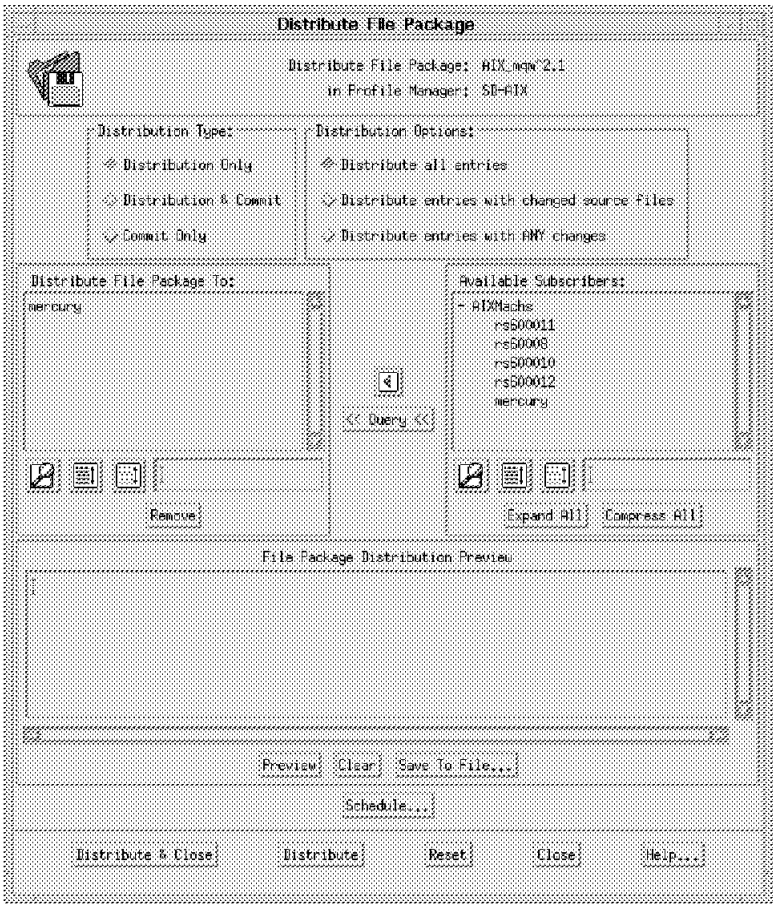


Figure 138. Distribute File Package (MQSeries with Subscriber)

7. Select **Distribute & Close** to begin distributing the file package to the target mercury.

 Note: The dialog will not be dismissed until the distribution is complete. If the distribution fails for any of the subscribers, a pop-up dialog is displayed to inform you which subscribers failed distribution.

 Now Courier will start the Before Distribution script specified in the file package. Then it will send from the source to the target any source directories and files specified.

 If the Before Distribution script completes successfully, then Courier will start the After Distribution script. If the Before Distribution exits with a return code other than zero, then Courier will stop the distribution.

Once the distribution is complete, you can check the distribution result by looking at the primary desktop panel, which will list the software distribution activities in the Operation Status field.

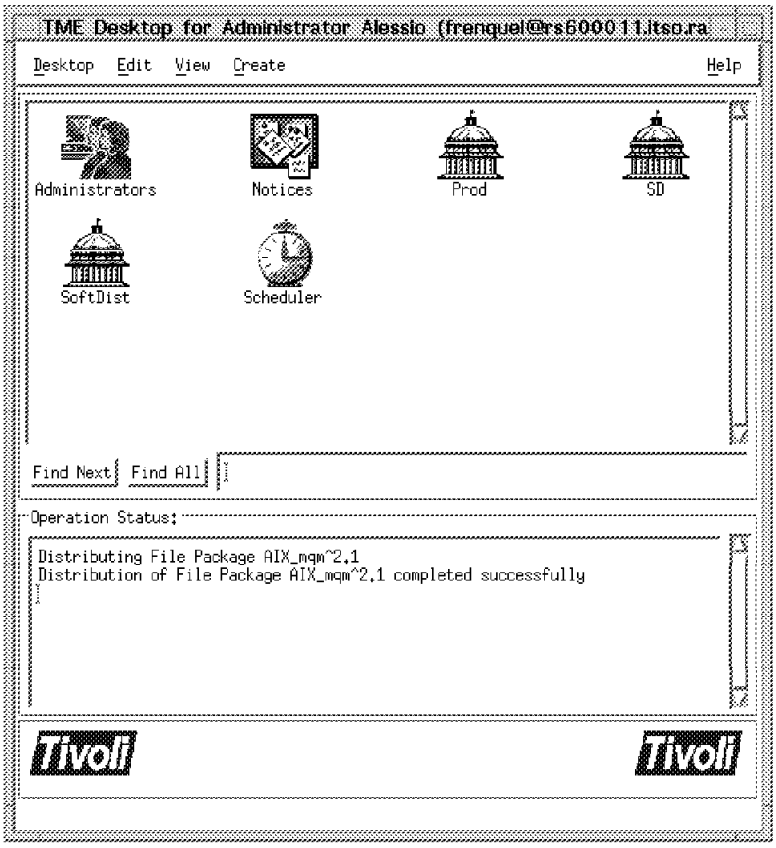


Figure 139. Desktop with Operation Status Messages

You also need to check the log file that we specified in the Log Information Options field when we created the file package properties in 5.4.3.3, “Create the File Package Properties” on page 151.

The file /courier/logs/aix/mqm.log on rs600011 will contain the following messages:

```
================
File Package:  "AIX_mqm^2.1"
Operation:     install  (m=1)
Finished:      Thu Aug 15 19:24:27 1996
--------
Source messages:
<none>
--------
mercury: SUCCESS
temp script: unix_before: /tmp/unix_VOs.Ma
temp script: unix_after: /tmp/unix_VEs.Mb
temp script: unix_removal: /tmp/unix_VHs.Mc
starting script: /tmp/unix_VEs.Ma
script complete: status=0
```

We have also specified in the File Package Properties panel the Send to Courier notice group option, so that the Courier Notice group will include notice messages for file package operations.

1. From the primary desktop select the **Notices** icon to display the Read Notices menu.



Figure 140. Read Notices

2. Select the **Courier** option as shown in Figure 140. Then select the **Notice** message as shown in Figure 141.

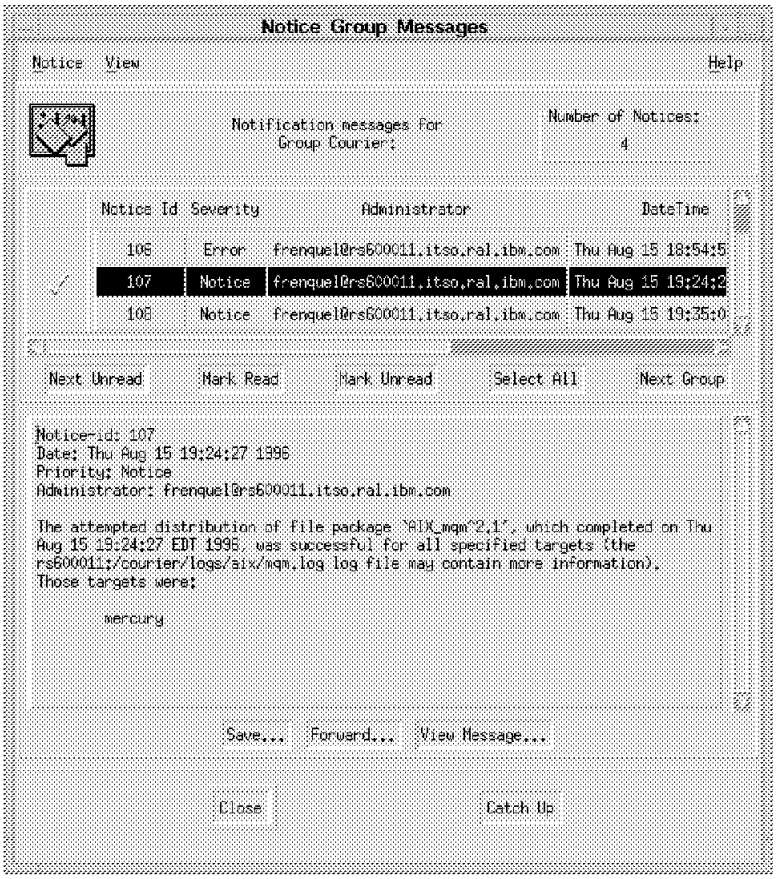


Figure 141. Notice Group Messages (Courier)

The mqm.sh script redirects the all script output to the /tmp/mqm.log file on the target machine. Therefore, you can check the script results by looking at the file on the target machine (in our case, mercury):

```
Installation Summary
--------------------
Name                 Level           Part        Event       Result
-------------------------------------------------------------------------------

mqm.samples          2.2.1.0         USR         APPLY       SUCCESS
mqm.runtime         2.2.1.0         USR         APPLY       SUCCESS
mqm.server           2.2.1.0         USR         APPLY       SUCCESS
mqm.dt_client        2.2.1.0         USR         APPLY       SUCCESS
mqm.books            2.2.1.0         USR         APPLY       SUCCESS
mqm.base             2.2.1.0         USR         APPLY       SUCCESS
mqm.aix_client       2.2.1.0         USR         APPLY       SUCCESS
mqm.Ja_JP            2.2.1.0         USR         APPLY       SUCCESS
mqm.Fr_FR            2.2.1.0         USR         APPLY       SUCCESS
mqm.Es_ES            2.2.1.0         USR         APPLY       SUCCESS
mqm.De_DE            2.2.1.0         USR         APPLY       SUCCESS
mqm.samples          2.2.1.0         ROOT        APPLY       SUCCESS
mqm.runtime          2.2.1.0         ROOT        APPLY       SUCCESS
mqm.server           2.2.1.0         ROOT        APPLY       SUCCESS
mqm.dt_client        2.2.1.0         ROOT        APPLY       SUCCESS
mqm.books            2.2.1.0         ROOT        APPLY       SUCCESS
mam.base             2.2.1.0         ROOT        APPLY       SUCCESS
mqm.aix_client       2.2.1.0         ROOT        APPLY       SUCCESS
mqm.Ja_JP            2.2.1.0         ROOT        APPLY       SUCCESS
mqm.Fr_FR            2.2.1.0         ROOT        APPLY       SUCCESS
mqm.Es_ES            2.2.1.0         ROOT        APPLY       SUCCESS
mqm.De_DE            2.2.1.0         ROOT        APPLY       SUCCESS
epw.lib              1.2.1.0         USR         APPLY       SUCCESS
epw.kext             1.2.1.0         USR         APPLY       SUCCESS
epw.doc              1.2.1.0         USR         APPLY       SUCCESS
epw.bin              1.2.1.0         USR         APPLY       SUCCESS
epw.adm              1.2.1.0         USR         APPLY       SUCCESS
dtext.brwsr.obj      2.3.0.1         USR         APPLY       SUCCESS
mqm.dtext_books      2.2.1.0         USR         APPLY       SUCCESS
mqm.dtext_books      2.2.1.0         ROOT        APPLY       SUCCESS
MQSeries queue manager created.
MQSeries queue manager started.
AMQ8006: MQSeries queue created.
AMQ8010: MQSeries process created.
AMQ8014: MQSeries channel created
0513-095 The request for subsystem refresh was completed successfully.
Installation completed successfully
```

5.4.4.2 Remove the File Package for MQSeries

After a file package has been distributed on a target, it can be removed using the Remove function. The remove operation will remove any files that were previously distributed and also trigger any Before or After distribution program.

Note that Courier does not perform any check if the package was previously distributed on the target since it does not have any *history* associated with the file package.

1. From the SoftDist policy region double-click on the **SD-AIX** profile manager.

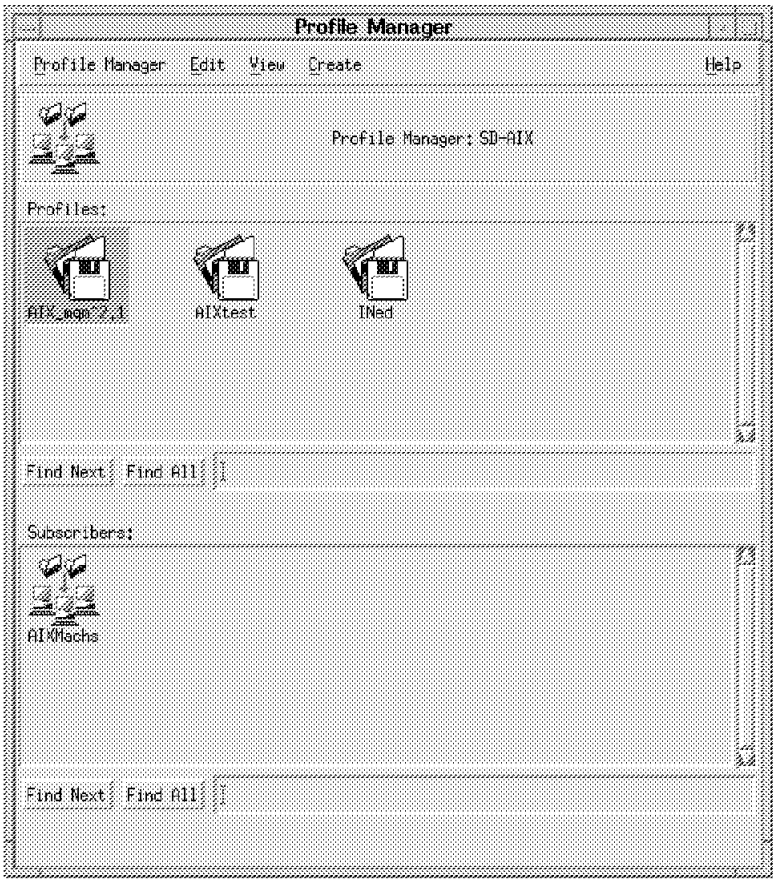


Figure 142. Profile Manager (SD-AIX)

2. Double-click on the **AIX_mqm^2.1** file package icon to display the panel shown in Figure 143 on page 169.

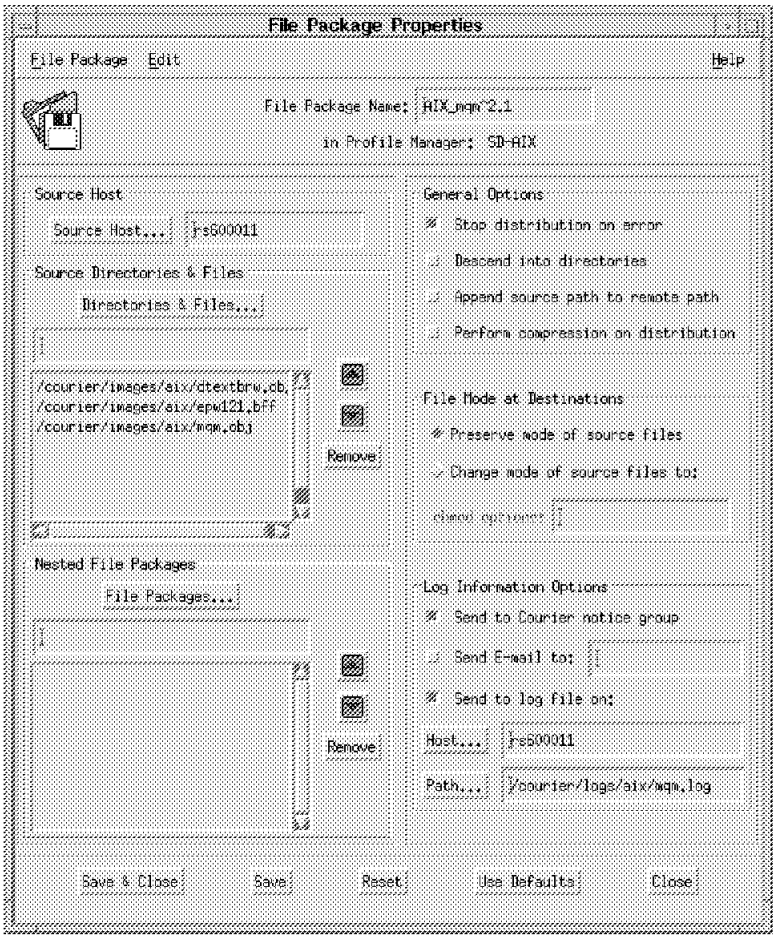


Figure 143. File Package Properties (MQSeries)

3. Select **File Package = > Remove from Hosts...** to display the following panel:

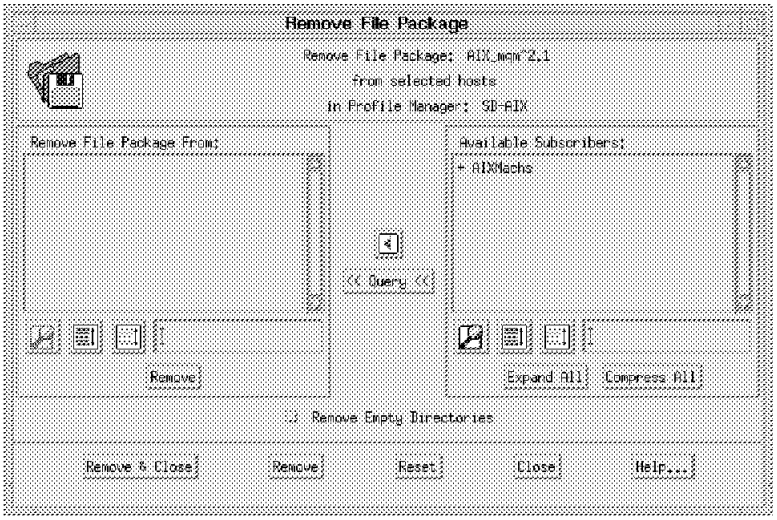


Figure 144. Remove File Package (MQSeries)

4. Fill in the Remove File Package From scrolling list with the subscribers from which you want to remove the file package.

You can choose the subscribers from the Available Subscribers list and move them to the Remove File Package From list using the arrow button.

In our case we have AIXMachs as the profile manager in the Available Subscribers list. Therefore, we double-click on the profile manager, which is prefixed by the character + to display all of the managed nodes that subscribe to the profile manager. We then select **mercury**.

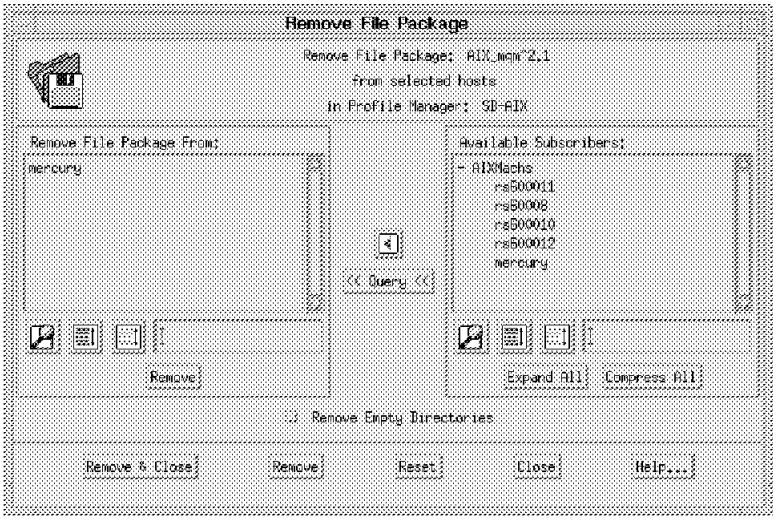


Figure 145. Remove File Package (MQSeries from Subscriber)

5. Select the **Remove & Close** push button to remove the file package from the subscriber mercury.

 Note: The dialog will not be dismissed until the removal is complete. If the removal fails for any of the subscribers, a pop-up dialog is displayed to inform you on which subscribers the remove failed.

 Now Courier will start the Upon Removal script specified in the file package and then will remove from the target any source directories and files previously distributed.

 Remove Options

 Using the Dialog you can only specify the Upon Removal script, which will be invoked only before the removal operation. If you want to run an after removal script, you need to export the file package, modify it, and then import it again. See 5.1.5, “Import/Export File Packages” on page 106 for more details.

Once the removal is complete, you can check the operation result by looking at the primary desktop panel, which will list the software distribution activities in the Operation Status field.

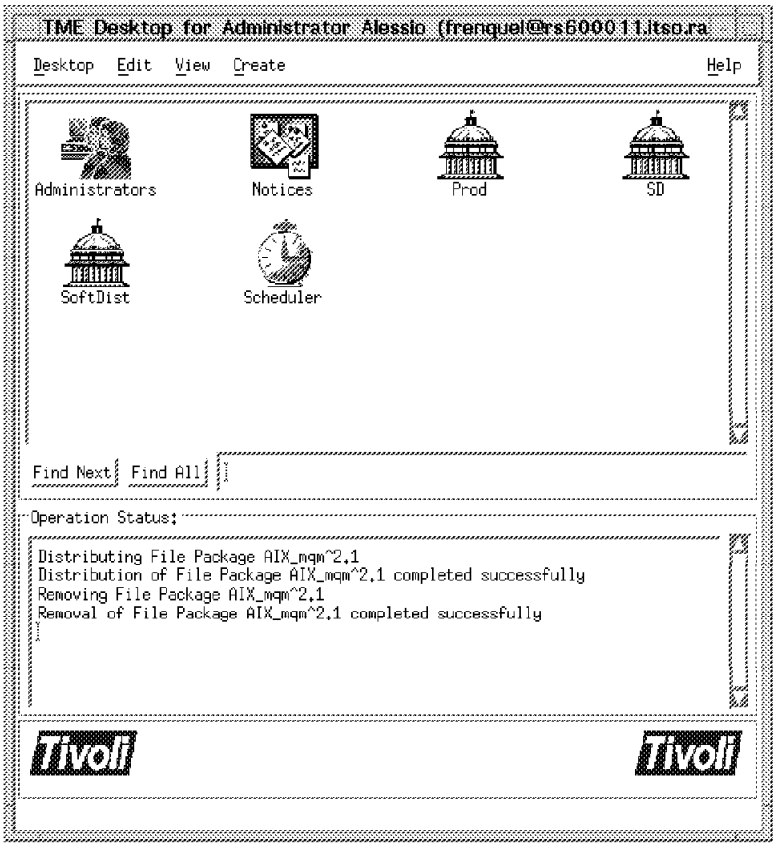


Figure 146. Desktop with Operation Status Messages

You also need to check the log file that we specified in the Log Information Options when we created the file package properties in 5.4.3.3, “Create the File Package Properties” on page 151.

The file /courier/logs/aix/mqm.log on rs600011 will contain the following messages:

```
================
File Package:  "AIX_mqm^2.1"
Operation:     uninstall (m=1)
Finished:      Thu Aug 15 19:35:00 1996
--------
Source messages:
<none>
--------
mercury: SUCCESS
temp script: unix_before: /tmp/unix_VOs.Ma
temp script: unix_after: /tmp/unix_VEs.Mb
temp script: unix_removal: /tmp/unix_VHs.Mc
starting script: /tmp/unix_VHs.Mc
script complete: status=0
```

We have also specified in the file Package properties the Send to Courier notice group option, so that the Courier notice group will include notice messages for file package operations.

The installation of this software is described in 5.4.4.1, “Distribute the File Package for MQSeries” on page 160:



Figure 147. Notice Group Messages (Courier)

Also the mqm.sh script redirects the all script output to the /tmp/mqm.log file on the target machine. Therefore, you can check the script results by looking at the file on the target machine (in our case, mercury).

```
Installation Summary
--------------------
Name                 Level           Part        Event       Result
-------------------------------------------------------------------------

epw.adm              1.2.1.0         USR         DEINSTALL   SUCCESS
epw.bin              1.2.1.0         USR         DEINSTALL   SUCCESS
epw.doc              1.2.1.0         USR         DEINSTALL   SUCCESS
epw.kext             1.2.1.0         USR         DEINSTALL   SUCCESS
epw.lib              1.2.1.0         USR         DEINSTALL   SUCCESS
mqm.De_DE            2.2.1.0         ROOT        DEINSTALL   SUCCESS
mqm.De_DE            2.2.1.0         USR         DEINSTALL   SUCCESS
mqm.Es_ES            2.2.1.0         ROOT        DEINSTALL   SUCCESS
mqm.Es_ES            2.2.1.0         USR         DEINSTALL   SUCCESS
mqm.Fr_FR            2.2.1.0         ROOT        DEINSTALL   SUCCESS
mqm.Fr_FR            2.2.1.0         USR         DEINSTALL   SUCCESS
mqm.Ja_JP            2.2.1.0         ROOT        DEINSTALL   SUCCESS
mqm.Ja_JP            2.2.1.0         USR         DEINSTALL   SUCCESS
mqw.aix_client       2.2.1.0         ROOT        DEINSTALL   SUCCESS
mqm.aix_client       2.2.1.0         USR         DEINSTALL   SUCCESS
mqm.base             2.2.1.0         ROOT        DEINSTALL   SUCCESS
mqm.base             2.2.1.0         USR         DEINSTALL   SUCCESS
mqm.books            2.2.1.0         ROOT        DEINSTALL   SUCCESS
mqm.books            2.2.1.0         USR         DEINSTALL   SUCCESS
mqm.dt_client        2.2.1.0         ROOT        DEINSTALL   SUCCESS
mqm.dt_client        2.2.1.0         USR         DEINSTALL   SUCCESS
mqm.dtext_books      2.2.1.0         ROOT        DEINSTALL   SUCCESS
mqm.dtext_books      2.2.1.0         USR         DEINSTALL   SUCCESS
mqm.samples          2.2.1.0         ROOT        DEINSTALL   SUCCESS
mqm.samples          2.2.1.0         USR         DEINSTALL   SUCCESS
mqm.server           2.2.1.0         ROOT        DEINSTALL   SUCCESS
mqm.server           2.2.1.0         USR         DEINSTALL   SUCCESS
dtext.brwsr.obj      2.3.0.1         USR         DEINSTALL   SUCCESS
mqm.runtime          2.2.1.0         ROOT        DEINSTALL   SUCCESS
mqm.runtime          2.2.1.0         USR         DEINSTALL   SUCCESS
Remove completed successfully
```

5.4.5 Create the MQSeries for NT File Package

The MQSeries for NT is an IBM product that provides application programming services that let application programs communicate with each other using *message queues*. For more information,refer to *MQSeries for Windows NT System Management Guide*, SC33-1643.

The MQSeries for NT package is available as a self-extracting EXE file.

5.4.5.1 Planning the Installation

One good practice is to identify all of the activities and actions needed to install the product and possibly group them into:

- Before Distribution
- After Distribution
- Commit
- Remove

First it will be necessary to distribute the self-extracting file to the NT machine. Then we will need to perform the following steps coded in the After Distribution script.

- Extract from the self-extracting file the directory structure of the MQSeries installable image, using the command:

  ```
  nt200cd -d
  ```

- Delete the self-extracting file to gain space.
- Invoke the unattended install using a response file, with the command:

  ```
  setup /r: C:\WTR05121.RSP
  ```

Once the MQSeries for NT is installed, we must initialize and set the MQSeries default objects. These steps will be coded in the Commit script:

- Create the queue manager, using the command:

  ```
  crtmqm /q QMNAME
  ```

- Start the default queue manager, using the command:

  ```
  strmqm QMNAME
  ```

- Create the system and default objects, using the command:

  ```
  runmqsc < C:\MQM\MQSC\AMQSCOMA.TST > DEFOBJ.OUT
  ```

To remove the MQSeries, some additional steps are necessary. These steps will be coded in the Upon Removal script:

- Stop the queue manager, using the command:

  ```
  endmqm QMNAME
  ```

- Wait and then invoke the Uninstall program, using the command:

  ```
  UNinstMQ /f
  ```

5.4.5.2 Create the Staging Area for the Source Files

rs600011 is used as the source host and the self-extracting file needed to install the MQSeries for NT is copied into:

```
/courier/images/nt/nt200cd.exe
```

Together with the self-extacting files, it will also be necessary to distribute the response file for the unattended installation. Some utilities for NT needed to run the configuration .BAT files. All of them are also stored in:

```
/courier/images/nt/WTR05121.bat
/courier/images/nt/sleep.exe
/courier/images/nt/win32gnu.dll
```

5.4.5.3 Create the File Package Properties

1. From the SoftDist policy region double-click on the **SD-AIX** profile manager previously created in 5.3.3.1, “Create Profile Managers to Group Nodes and File Packages” on page 113.

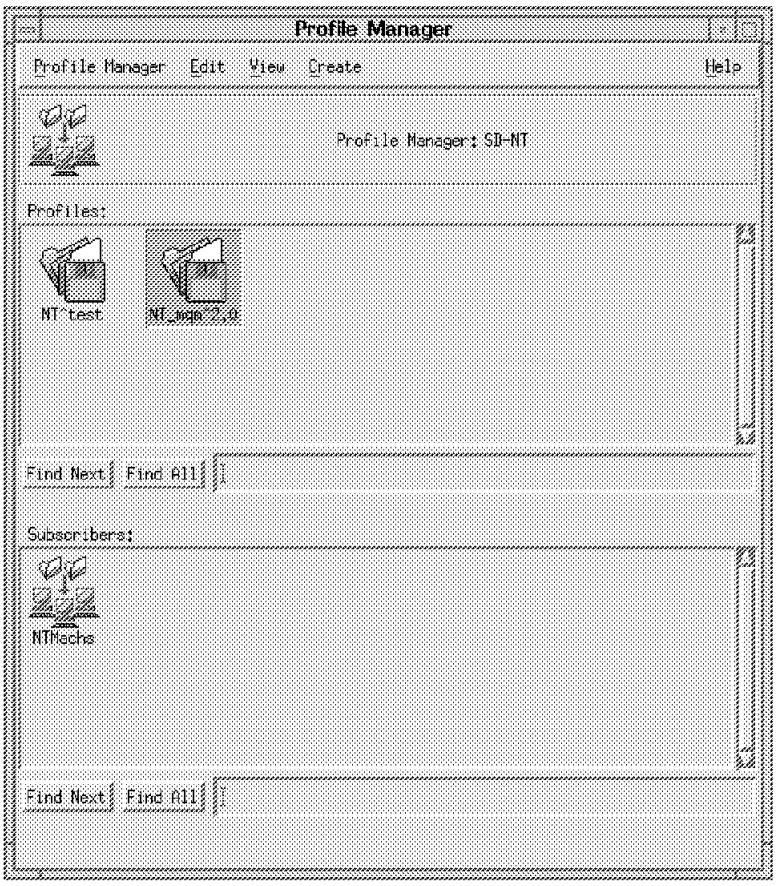


Figure 148. Profile Manager (SD-NT)

2. Double-click on the **NT_mqm^2.0** file package. The TME displays the following dialog:

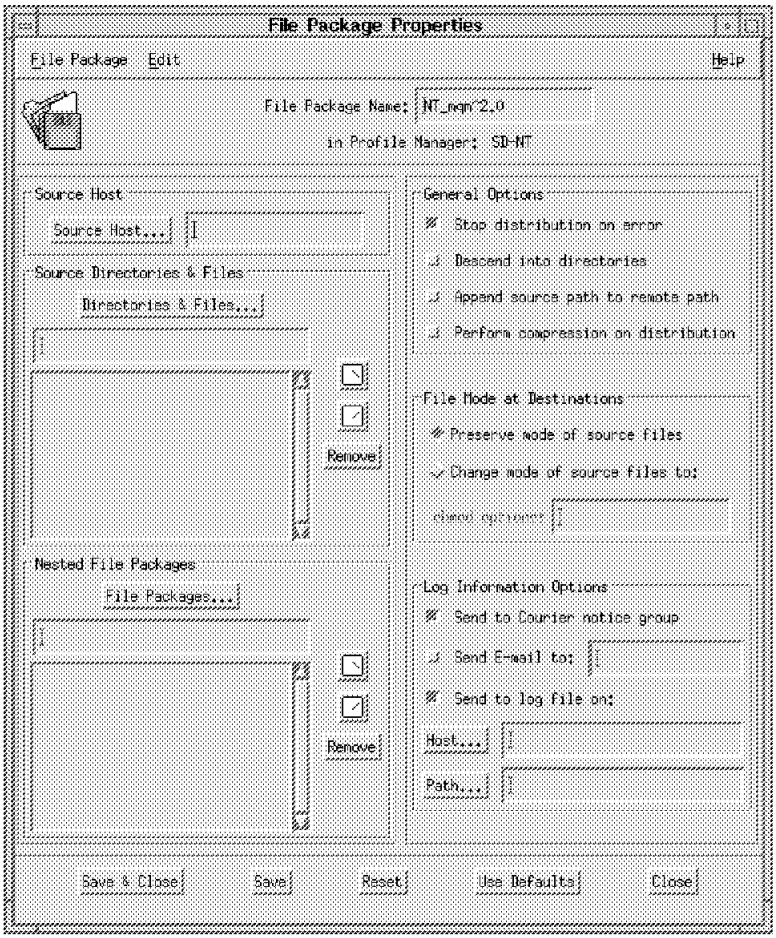


Figure 149. File Package Properties (MQSeries NT)

The File Package Properties panel shows the file package name, the directories and files, and nested file packages to be included.

3. Set the Source Host of the file package.

 Select the **Source Host...** option to identify the name of the host on which the file package source files resides (in our case, rs600011).

 Source Host

 Only UNIX managed nodes are supported as source hosts.

4. Set the Source Directories & Files of the file package.

 Select the **Directories & Files...** option to identify the full path of the source files. In our case, they were:

   ```
   /courier/images/nt/nt200cd.exe
   /courier/images/nt/sleep.exe
   /courier/images/nt/win32gnu.dll
   /courier/images/nt/WTR05121.rsp
   ```

5. Set the Log Information Options to control the logging activity.

 Select the **Send to Courier notice group** check box to have Courier post a notice, which includes an indication of success or failure of the operation for each target, to the Courier notice group when a file package operation occurs.

Select the **Send to log file on:** check box to have Courier place log information in the specified file when a file package operation occurs. An entry is made to the log each time a file package is distributed, committed or removed.

In our case we defined the host to be rs600011 and the path to be /courier/logs/nt/mqm.log.

Log

It is very important to set the **Send to log file on** check box. Otherwise, vital information regarding the distribution operations will never be logged.

6. We then left the defaults in all of the other boxes.

 The File Package Properties window will now look as follows:

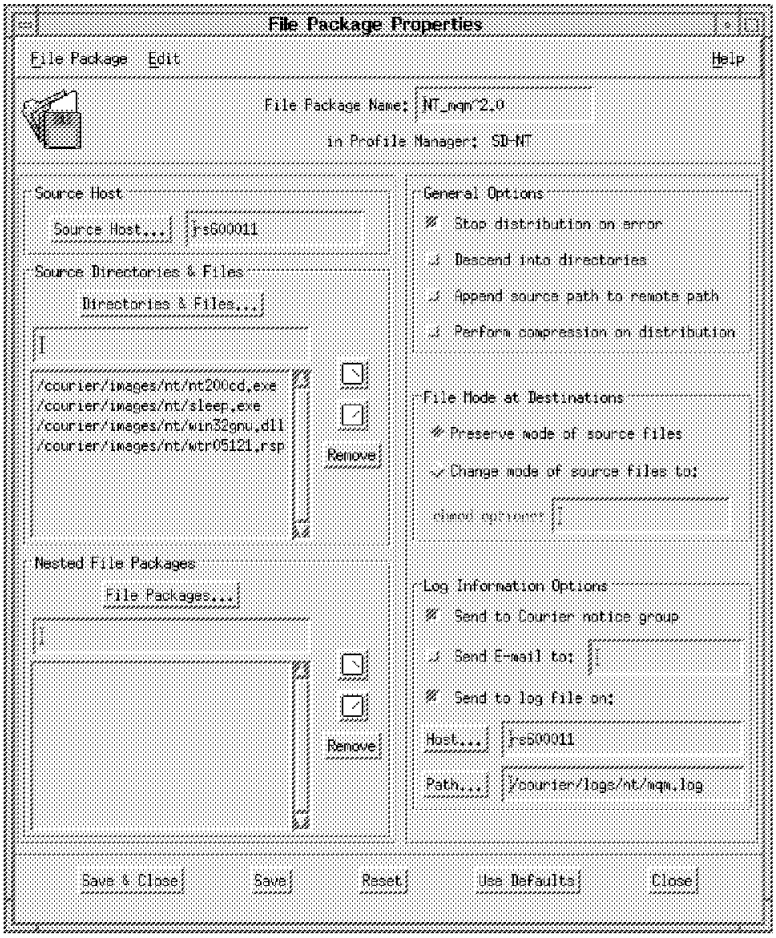


Figure 150. File Package Properties (MQSeries NT)

5.4.5.4 Create the File Package Platform-Specific Options

After defining the file package properties, define the platform-specific options of the file package. In our example, we defined Windows NT file package options.

1. From the File Package Properties window shown in Figure 126 on page 152, select **Edit => Platform-Specific Options => Windows NT Options...** to display the following panel:

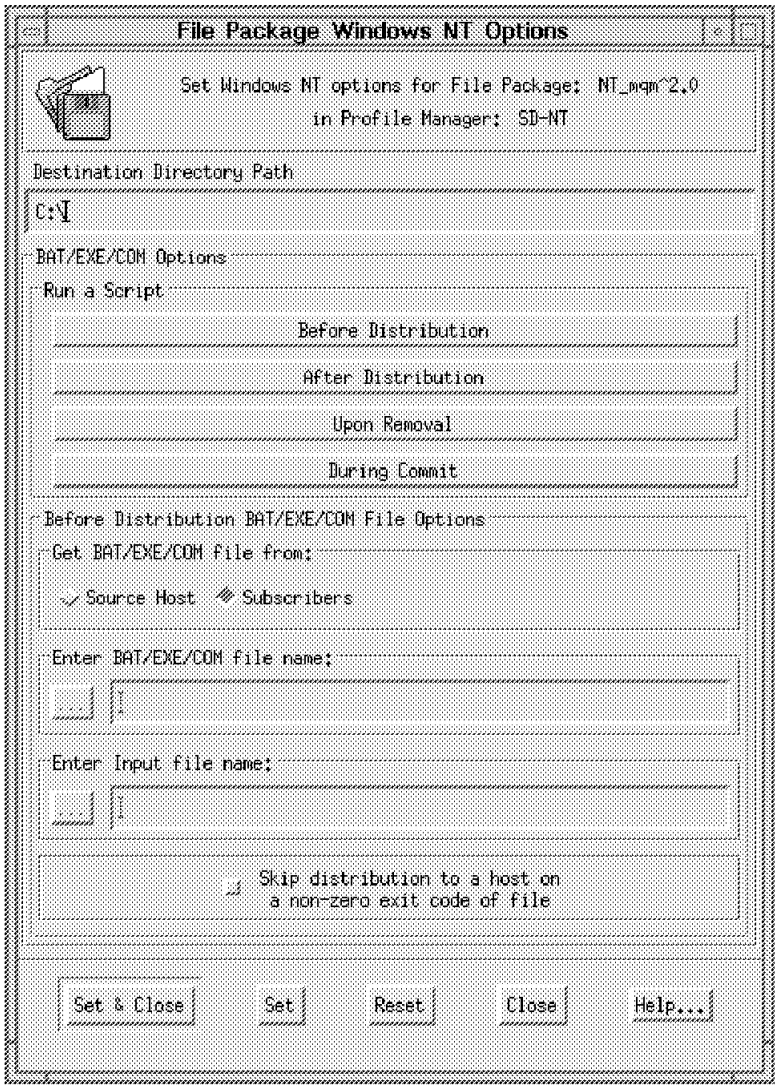


Figure 151. File Package Windows NT Options (MQSeries Before)

2. Set the Destination Directory Path to the directory full path of the target to which the file package is distributed. In our case we specified:

 C:\

3. Set the BAT/EXE/COM Options using the Run a Script buttons.

 These buttons display options that enable you to run configuration programs, which includes .BAT and .EXE files on subscribers before or after distributing, removing or committing a file package.

 The MQSeries for NT package needs the following program to be run:

 After Distribution This is a .BAT to extract the directory from the self-extracting file and run the unattended installation using a response file.

 During Commit This is a .BAT to configure the Queue Manager environment.

 Upon Removal This is a .BAT to end the Queue Manager and Uninstall the product.

4. Now we need to set the After Distribution option.

5. When you select the **After Distribution** push button, the following dialog is displayed:

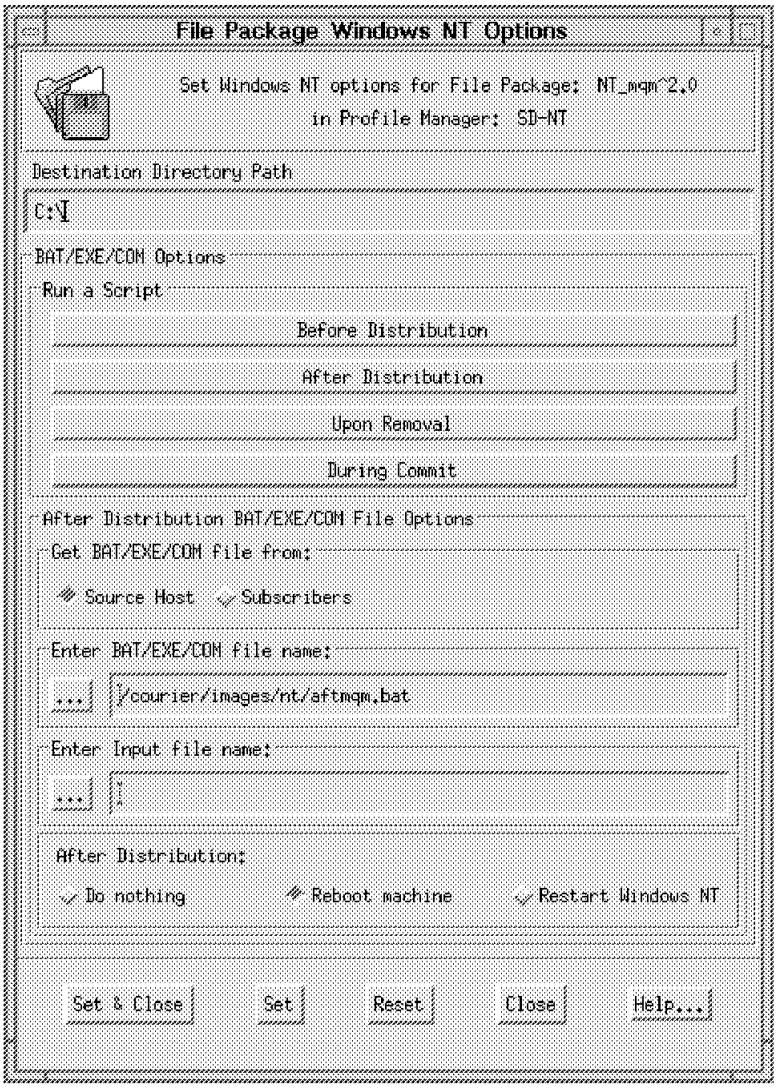


Figure 152. File Package Windows NT Options (MQSeries After)

6. Set the Get /BAT/EXE/COM file from field to specify the location of the program or procedure.

 Select the **Source Host** radio button to instruct Courier to copy the After Distribution program from the source host to a temporary file on each subscriber. It will then run the programs on each subscriber after the distribution and remove each temporary file from each subscriber upon completion.

7. Set the Enter /BAT/EXE/COM file name field to the full path of the program to run After Distribution. In our case it was:

   ```
   /courier/images/nt/aftmqm.bat
   ```

 See 5.4.5.5, “Create Program Options” on page 182 for a detailed description of this .BAT.

8. Select the **Reboot Machine** button on the bottom part of the panel because MQSeries for NT requires you to shut down and restart the computer after the installation.
9. When you select the **Upon Removal** button, the following dialog will be displayed:

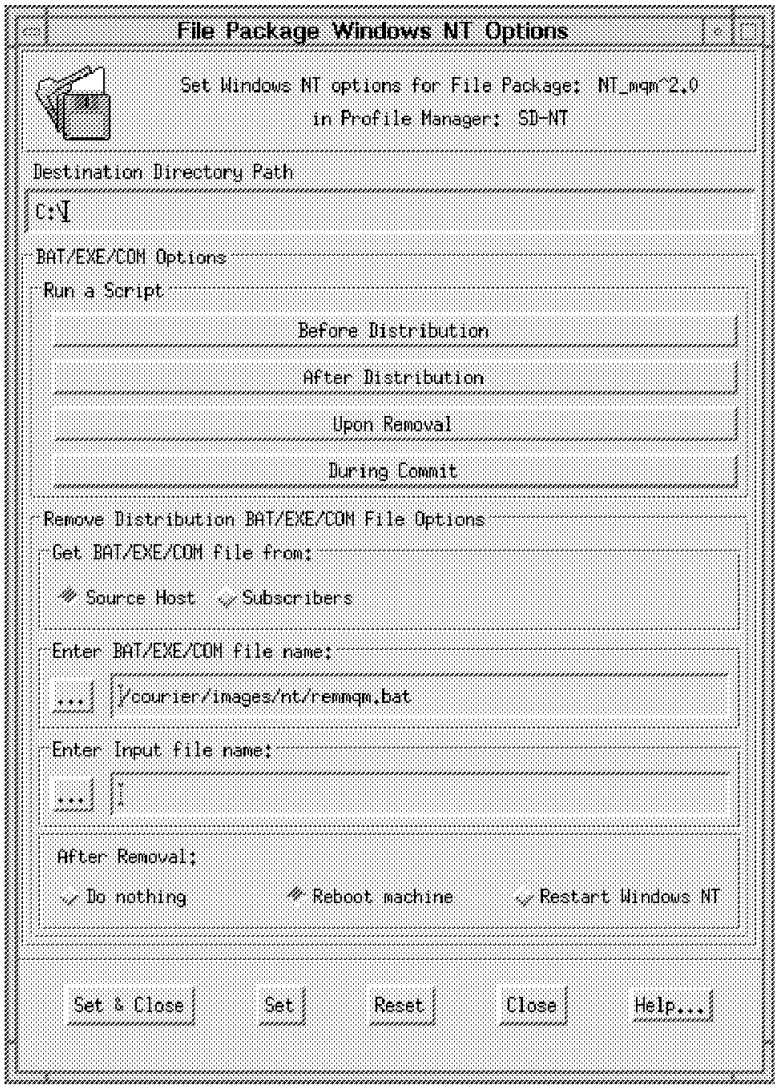


Figure 153. File Package Windows NT Options (MQSeries Remove)

10. Set the Get BAT/EXE/COM file from field to specify the location of the program or procedure.

 Select the **Source Host** radio button to instruct Courier to copy the Upon Removal program from the source host to a temporary file on each subscriber. It will then run the programs on each subscriber to perform the remove and remove each temporary file from each subscriber upon completion.
11. Set the Enter BAT/EXE/COM file name field to the full path of the program to run upon removal. In our case it is:

    ```
    /courier/images/nt/remmqm.bat
    ```

See 5.4.5.5, “Create Program Options” on page 182 for a detailed description of this .BAT.

12. Select the **Reboot Machine** button on the bottom part of the panel because MQSeries for NT requires you to shut down and restart the computer after the Uninstall is completed.

13. When you select the **During Commit** button, the following dialog is displayed:

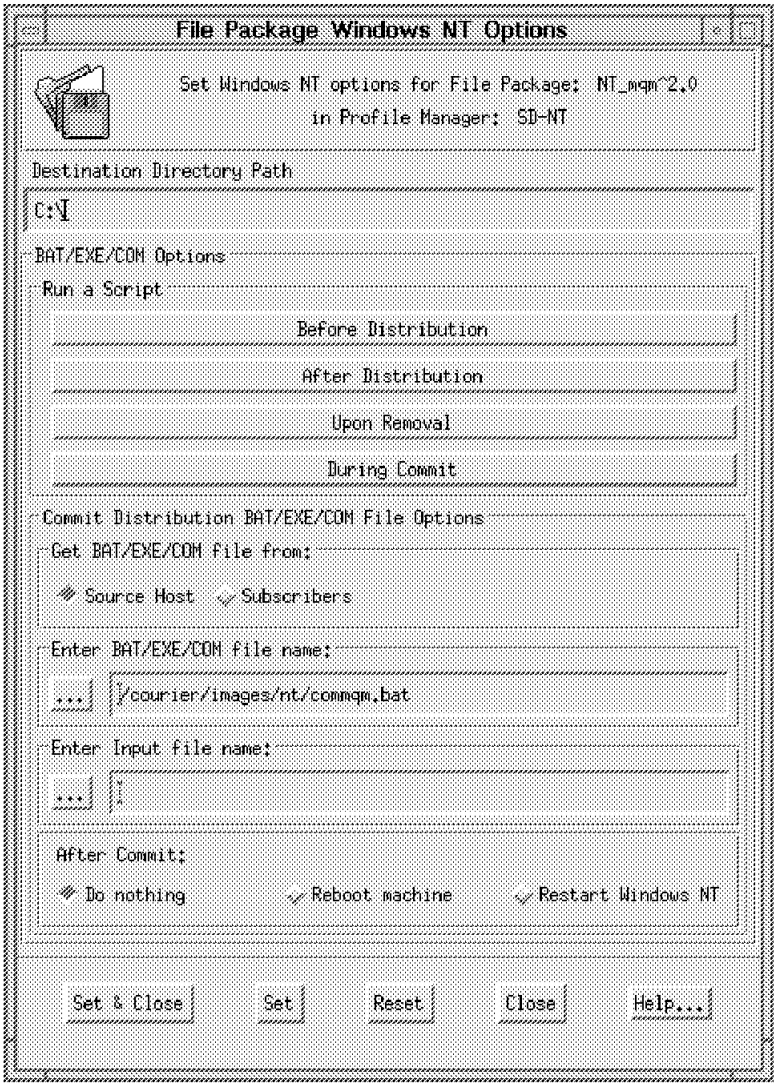


Figure 154. File Package Windows NT (MQSeries Commit)

14. Set the Get BAT/EXE/COM file from field to specify the location of the program or procedure.

 Select the **Source Host** radio button to instruct Courier to copy the During Commit program from the source host to a temporary file on each subscriber. It will then run the programs on each subscriber to perform the commit and remove each temporary file from each subscriber upon completion.

15. Set the Enter BAT/EXE/COM file name field to the full path of the program to run during commit. In our case it is:

    ```
    /courier/images/nt/commqm.bat
    ```

See 5.4.5.5, “Create Program Options” on page 182 for a detailed description of this .BAT.

Programs existence

Courier does not check for the existence of any of those programs on either the source host or subscribers until the time of software distribution. If a program is not found at that time, then an error will be logged.

16. Select the **Set & Close** button to apply all of the changes and close the File Package Windows NT Option window.
17. Select the **Save & Close** button from the File Package Properties window to apply all of the changes to the file package.

5.4.5.5 Create Program Options

After having defined the file package properties, it is necessary to create the program options. See 5.4.3, “Create the MQSeries for AIX File Package” on page 149 for a complete description of the requirements for the program options.

Those scripts will be stored in the source host rs600011 under the directory:

```
/courier/images/nt
```

In order to install the MQSeries for NT, we need to extract from the self-extracting .exe file the directory structure. Then delete the self-extracting file to gain space and install the product using the unattended installation method. This will be coded in the After Installation program.

Once the installation is complete, we then need to initialize the MQSeries for NT environment. To do so we will use a Commit program and finally we will code a Remove program to uninstall the MQSeries, as follows:

1. aftmqm.bat - To Install the MQSeries package
2. commqm.bat - To Commit the MQSeries package
3. remmqm.bat - To Remove the MQSeries package

Figure 155 gives a list of the .BAT files:

```
c:
cd c:\
nt200cd -d
del nt200cd.exe
c:\setup\En_US\setup /r: c:\wtr05121.rsp
```

Figure 155. The MQSeries NT aftmqm.bat File

```
c:
cd c:\
crtmqm /q QMNAME
sleep 10
strmqm QMNAME
sleep 10
runmqsc < c:\mqm\mqsc\amqscoma.tst > defobj.out
sleep 10
del c:\image\*.* /q
rmdir  image
```

Figure 156. The MQSeries NT commqm.bat File

```
c:
cd c:\
endmqm QMNAME
sleep 60
UninstMQ /f
```

Figure 157. The MQSeries NT remmqm.bat File

Following is also the response file that we used to install the MQSeries in unattended mode.

```
************************************************************************
*                                                                      *
*          MQSeries for Windows NT from IBM                            *
*          Sample installation response file.                          *
*                                                                      *
* This is a sample installation response file which can be used        *
* in conjunction with SETUP.EXE (the program used to install           *
* MQSeries) to perform unattended installation of components of        *
* MQSeries.                                                            *
************************************************************************

************************************************************************
* SOURCE                                                               *
*                                                                      *
* Specifies the path for the file directory where the installation     *
* image resides.                                                       *
*                                                                      *
************************************************************************
SOURCE=c:\image

************************************************************************
* FILE                                                                 *
*                                                                      *
* Specifies the path for the file directory where the code for         *
* MQSeries will be installed.                                          *
*                                                                      *
************************************************************************
FILE=c:\mqm

************************************************************************
* WORK                                                                 *
*                                                                      *
```

```
* Specifies the path for the work directory where data files for    *
* MQSeries will be stored.                                          *
*                                                                   *
*********************************************************************
WORK=c:\mqm

*********************************************************************
* COMP                                                              *
*                                                                   *
* Specifies the name of an installable component of MQSeries.       *
*                                                                   *
* Valid parameters:                                                 *
*                                                                   *
*      Base product and Server                                      *
*      Desktop Clients                                              *
*      Online Information                                           *
*      Toolkit                                                      *
*      Windows NT Client                                            *
*                                                                   *
*********************************************************************
  COMP=Base product and Server
* COMP=Desktop Clients
* COMP=Online Information
* COMP=Toolkit
  COMP=Windows NT Client

*********************************************************************
* CFGUPDATE                                                         *
*                                                                   *
* Specifies whether the registry is automatically updated.          *
*                                                                   *
* Valid parameters:                                                 *
*                                                                   *
*  AUTO    automatically updates the registry                       *
*  MANUAL  does not update the registry                             *
*                                                                   *
*********************************************************************
CFGUPDATE=AUTO

*********************************************************************
* OVERWRITE                                                         *
*                                                                   *
* Specifies whether to overwrite files automatically during         *
* installation.                                                     *
*                                                                   *
* Valid parameters:                                                 *
*                                                                   *
* YES                                                               *
* NO                                                                *
*                                                                   *
*********************************************************************
OVERWRITE=NO
```

5.4.6 Software Distribution of MQSeries for NT

You can now distribute the MQSeries for NT file package to its subscribers. This chapter describes how to:

- Distribute MQSeries from rs600011 to target WTR05121
- Commit MQSeries on target WTR05121
- Remove MQSeries from target WTR05121

WTR05121 is a PC managed node and is managed by the rs60008 server. Both of those nodes belong to TMR SD. Please refer to 2.4.5, “Installing PC Managed Node as TME Client” on page 31 on how to install and configure a PC managed node.

We will start the Distribution from the TMR Prod whose server is rs600011. The MQSeries for NT file package will be distributed through the path shown in the following picture:

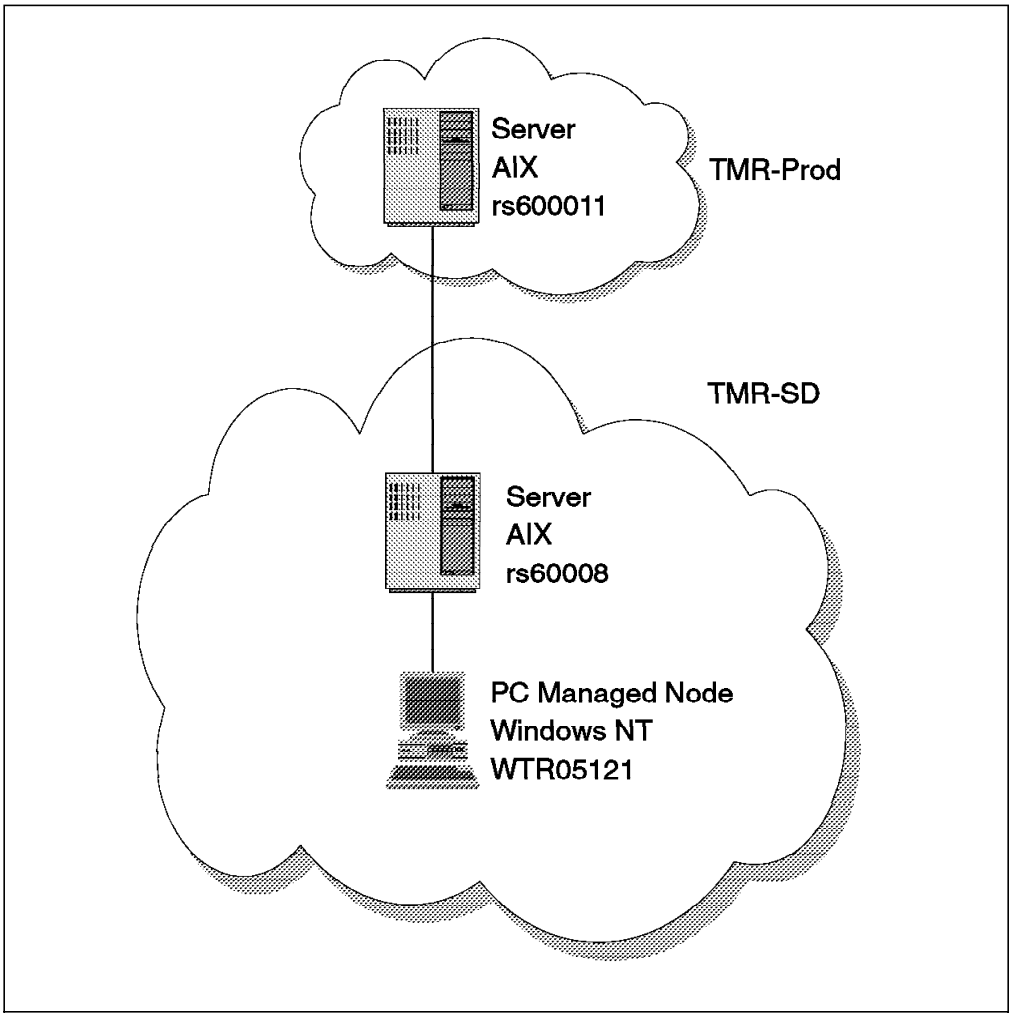


Figure 158. Distribution Path for the MQSeries for NT File Package

5.4.6.1 Distribute the File Package for MQSeries NT

To distribute a file package you can use *drag and drop*, dragging the file package icon and dropping it into a managed node or a profile manager, the command line or the desktop.

We show a scenario using the desktop:

1. From the SoftDist policy region double-click on the **SD-NT** profile manager.

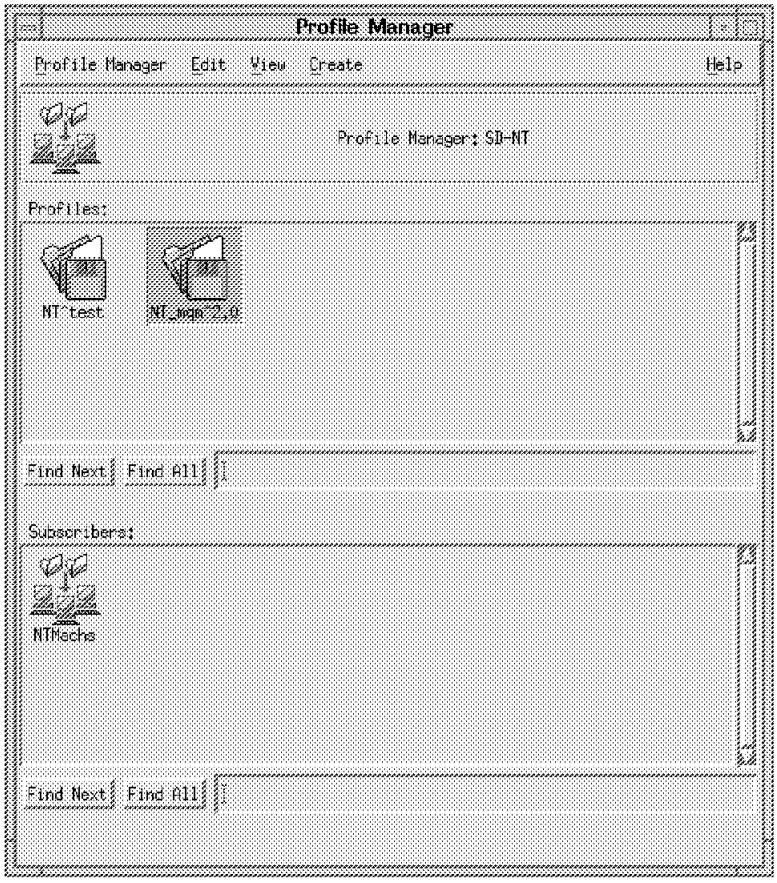


Figure 159. Profile Manager (SD-NT)

2. Double-click on the **NT_mqm^2.0** file package icon to display the panel shown in Figure 160 on page 187.

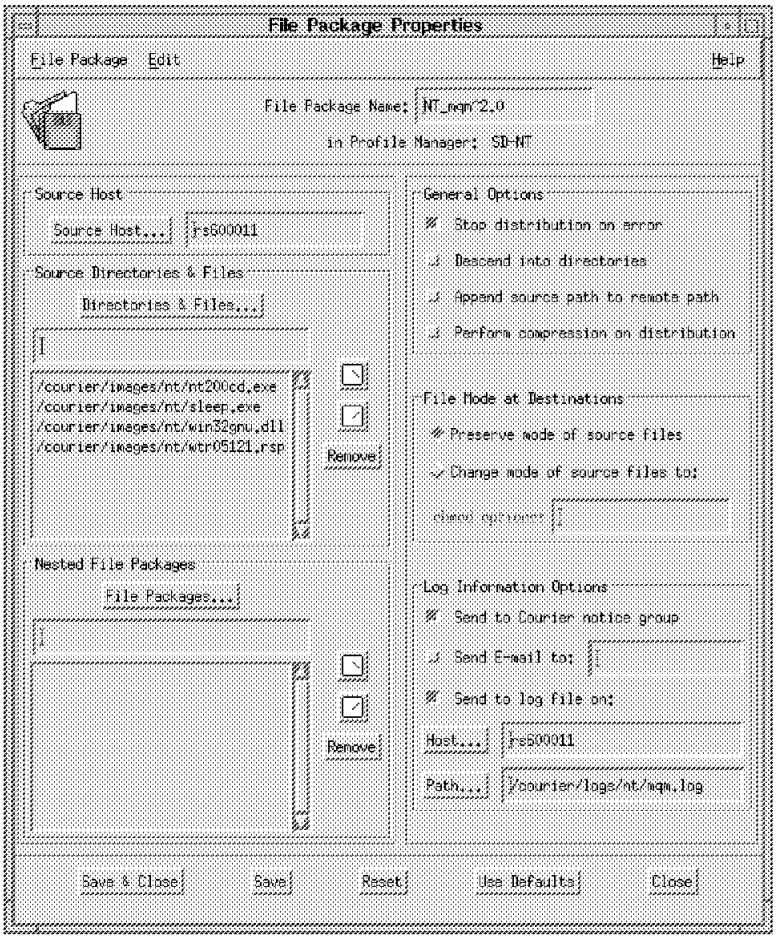


Figure 160. File Package Properties (MQSeries NT)

3. Select **File Package = > Distribute...** to display the following panel:

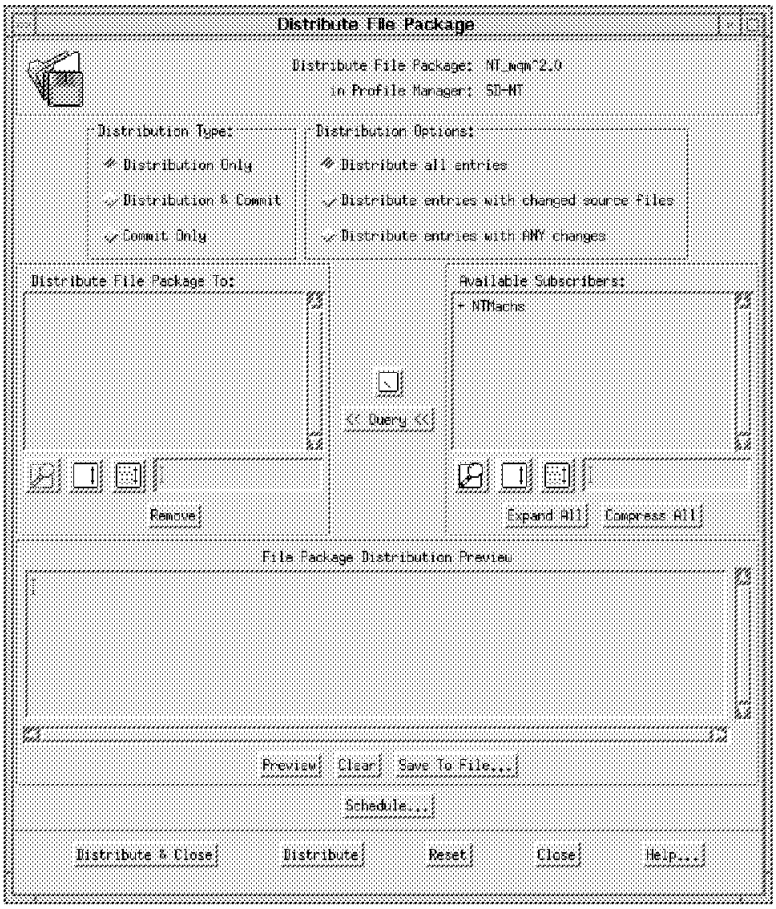


Figure 161. Distribute File Package (MQSeries NT)

4. Set the Distribution Type from the available list.

 Select **Distribution Only** to distribute only the file package. No specified commit programs are run during this operation.

5. Set the Distribution Options to control which files in the file package are distributed.

 Select **Distribute all entries** to distribute all files and directories in the file package.

6. Fill in the Distribute File Package To scrolling list with the subscribers to which you want to distribute the file package.

 You can choose the subscribers from the Available Subscribers list and move them to the Distribute File Package To list using the arrow button.

 In our case we have NTMachs as the profile manager in the Available Subscribers list. Therefore, we double-click on the profile manager, which is prefixed by the character + to display all of the managed nodes that subscribe to the profile manager. We then select **WTR05121**.

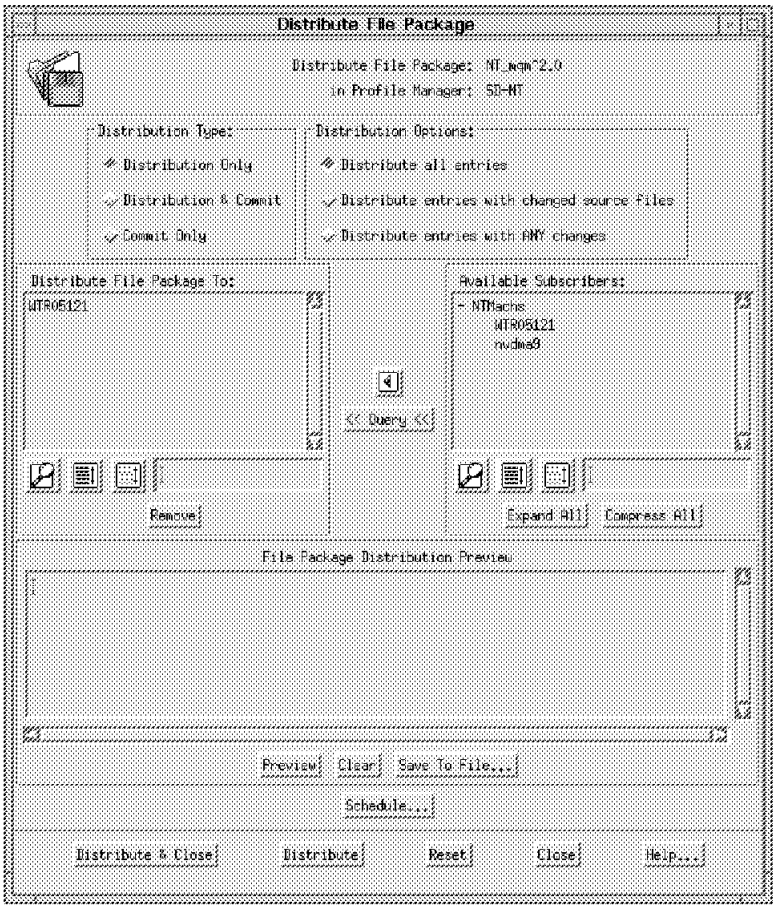


Figure 162. Distribute File Package (MQSeries NT with Subscriber)

7. Select the **Distribute & Close** button to begin distributing the file package to target WTR05121.

 Note: The dialog will not be dismissed until the distribution is complete. If the distribution fails for any of the subscribers, a pop-up dialog is displayed to inform you which subscribers failed distribution.

 Now Courier will send from the source to the target any source directories and files specified and then it will start the After Distribution program specified in the file package.

Once the distribution is complete, you can check the distribution result by looking at the primary desktop panel, which will list the software distribution activities in the Operation Status field.

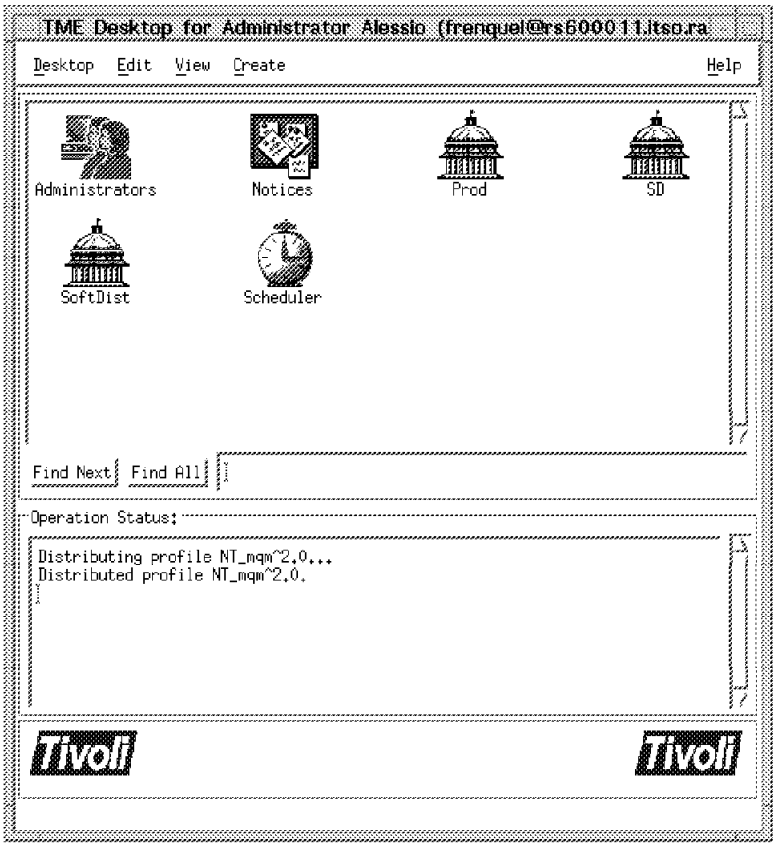


Figure 163. Desktop with Operation Status Messages

You also need to check the log file that we specified in the Log Information Options when we created the file package properties in 5.4.5.3, “Create the File Package Properties” on page 174.

The file /courier/logs/nt/mqm.log on rs600011 will contain the following messages:

```
================
File Package:  "NT_mqm^2.0"
Operation:     install  (m=6)
Finished:      Mon Aug 19 16:32:20 1996
--------
Source messages:
<none>
--------
WTRO5121: SUCCESS
temp script: nt_after: C:\TEMP\nt_a2.bat
temp script: nt_commit: C:\TEMP\nt_c3.bat
temp script: nt_removal: C:\TEMP\nt_r4.bat
starting script: C:\TEMP\nt_a2.bat
'C:\TEMP\nt_a2.bat' script complete: status=0
```

We have also specified in the File Package Properties panel the Send to Courier notice group option, so that the Courier notice group will include notice messages for file package operations.

From the Primary Desktop select the **Notices** icon to display the Read Notices menu. Then select the **Courier** option and the Notice message:

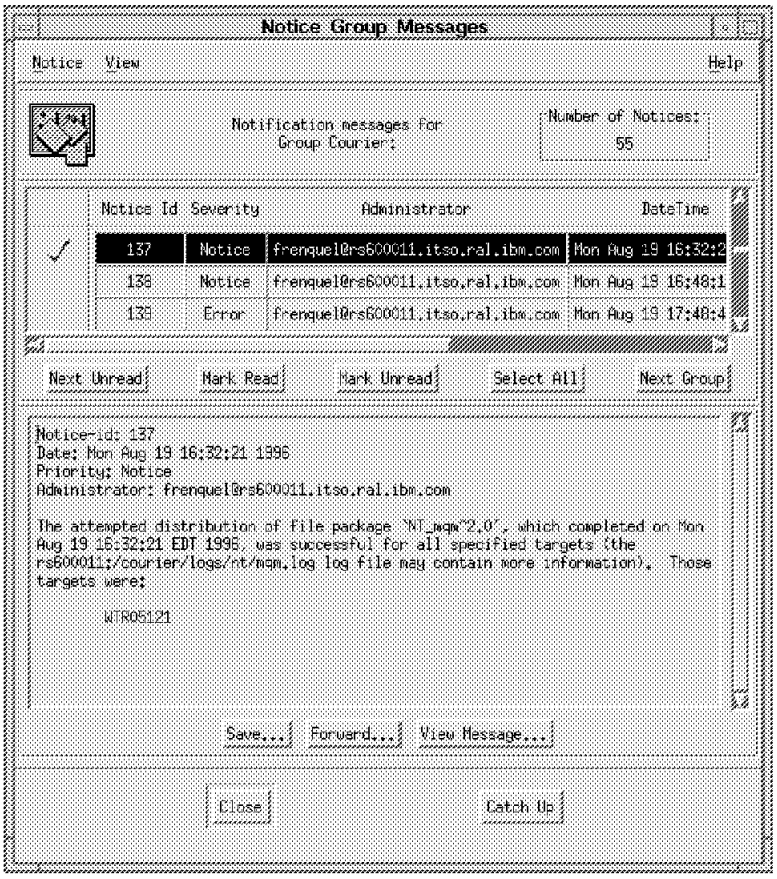


Figure 164. Notice Group Messages (Courier)

If you observe the NT server panel, it will display the MQSeries Installation panel in progress. Once the installation is completed, it will reboot as instructed by Courier with the **Reboot machine After Distribution** button set in 5.4.5.4, “Create the File Package Platform-Specific Options” on page 177.

5.4.6.2 Commit the File Package for MQSeries NT

The Commit operation will only trigger the During Commit program contained in the file package in the specified target. It is important to clarify that the commit operation is nothing else than executing the Commit program. Note that Courier does not perform any check if the package was previously distributed on the target since it does not have any *history* associated with the file package.

1. From the SoftDist policy region double-click on the **SD-NT** profile manager.

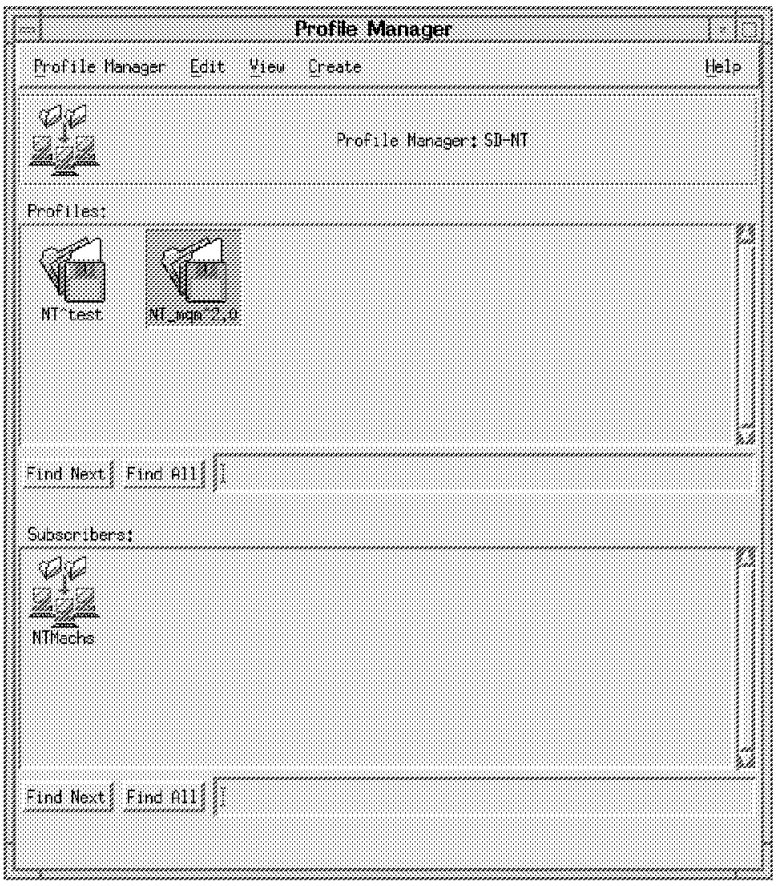


Figure 165. Profile Manager (SD-NT)

2. Double-click on the **NT_mqm^2.0** file package icon to display the panel shown in Figure 166 on page 193.

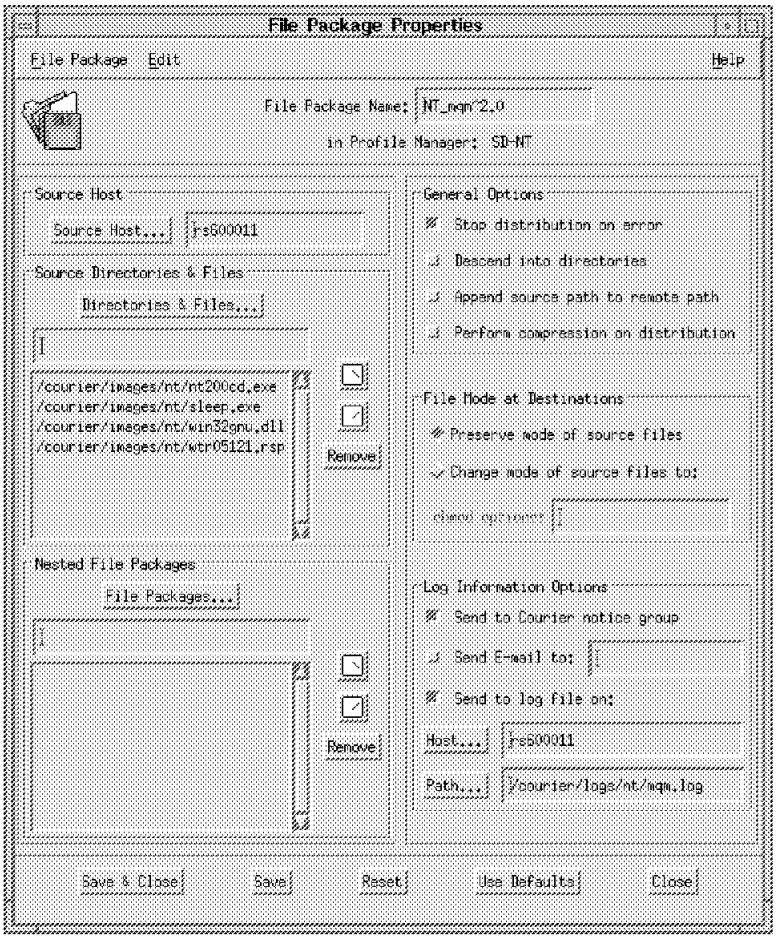


Figure 166. File Package Properties (MQSeries NT)

3. Select **File Package = > Distribute...** to display the following panel:

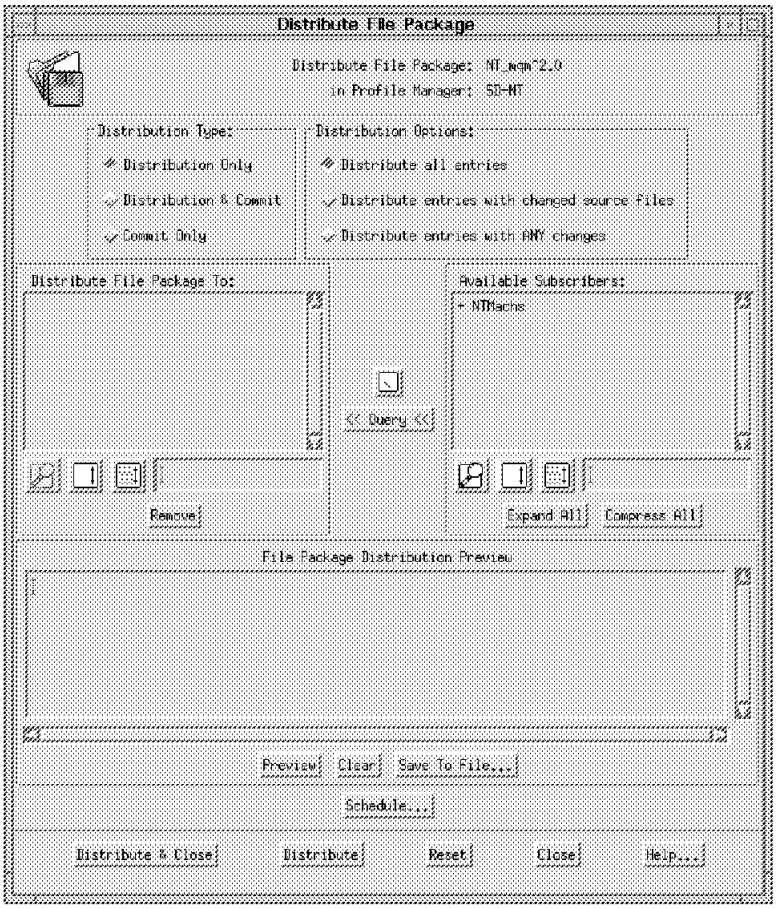


Figure 167. Distribute File Package (MQSeries NT)

4. Set the Distribution Type from the available list.

 Select **Commit Only** to run only the specified commit program on each subscriber.

5. Fill in the Distribute File Package To scrolling list with the subscribers on which you want to commit the file package.

 You can choose the subscribers from the Available Subscribers list and move them to the Distribute File Package To list using the arrow button.

 In our case we have NTMachs as the profile manager in the Available Subscribers list. Therefore, we double-click on the profile manager, which is prefixed by the character + to display all of the managed nodes that subscribe to the profile manager. We then select **WTR05121**.

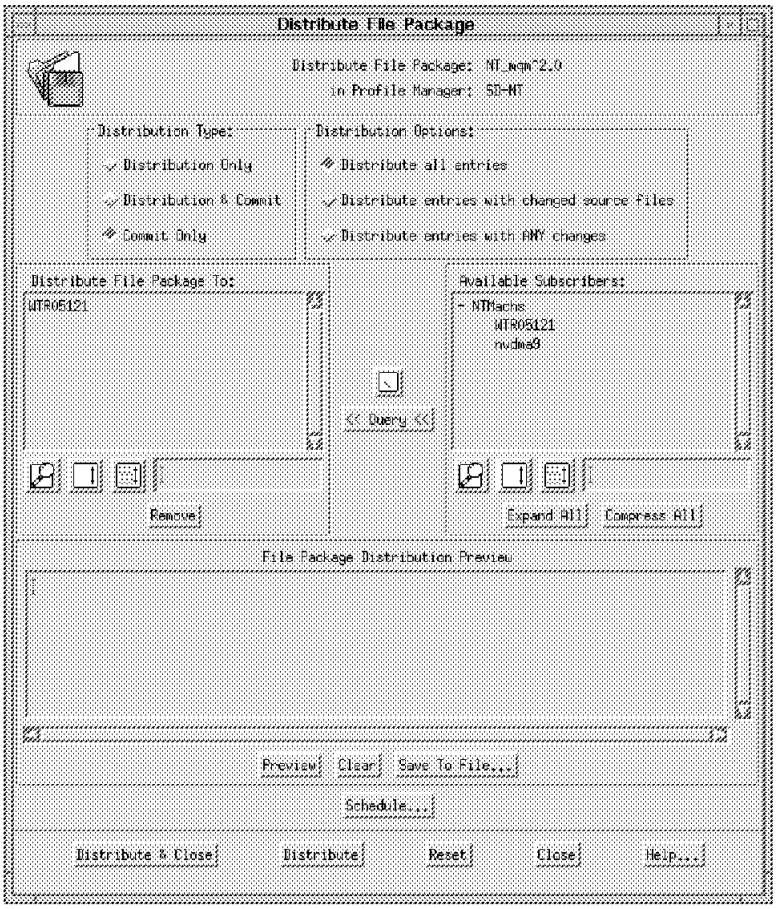


Figure 168. Commit File Package (MQSeries NT with Subscriber)

6. Select the **Distribute & Close** button to commit the file package on the target WTR05121.

 Note: The dialog will not be dismissed until the commit is complete. If the commit fails for any of the subscribers, a pop-up dialog is displayed to inform you which subscribers failed distribution.

 Now Courier will start the During Commit script specified in the file package.

Once the commit is complete, you can check the operation result by looking at the primary desktop panel, which will list the software distribution activities in the Operation Status field.

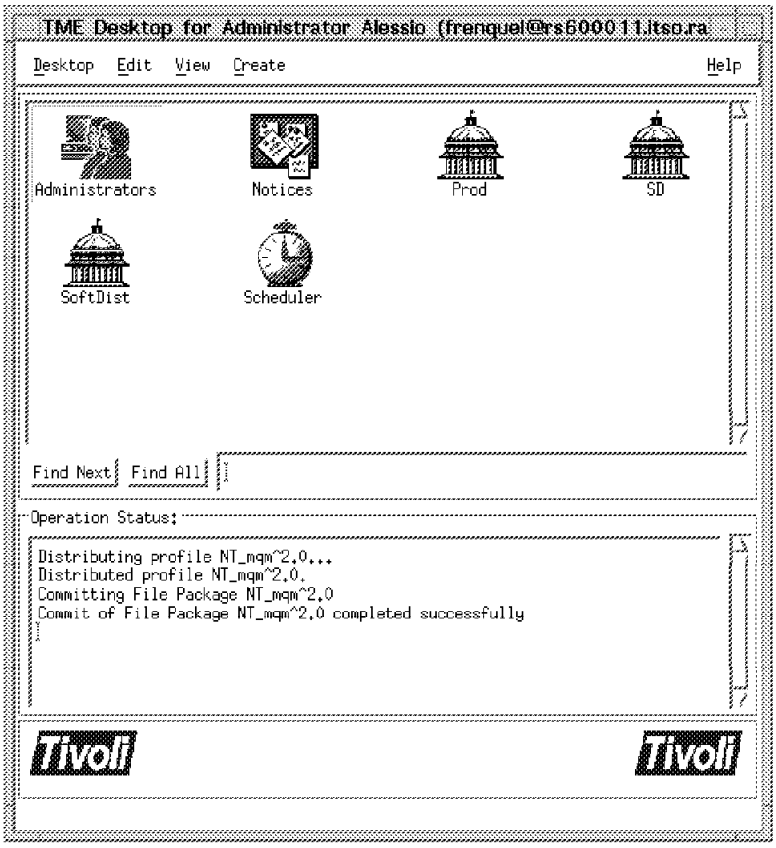


Figure 169. Desktop with Operation Status Messages

You also need to check the log file that we specified in the Log Information Options when we created the File package properties in 5.4.5.3, “Create the File Package Properties” on page 174.

The file /courier/logs/nt/mqm.log on rs600011 will contain the following messages:

```
================
File Package:  "NT_mqm^2.0"
Operation:     install  (m=6)
Finished:      Mon Aug 19 16:48:16 1996
--------
Source messages:
<none>
--------
WTRO5121: SUCCESS
temp script: nt_after: C:\TEMP\nt_a2.bat
temp script: nt_commit: C:\TEMP\nt_c3.bat
temp script: nt_removal: C:\TEMP\nt_r4.bat
starting script: C:\TEMP\nt_c3.bat
'C:\TEMP\nt_c3.bat' script complete: status=0
```

We have also specified in the File Package Properties panel the Send to Courier notice group option, so that the Courier notice group will include notice messages for file package operations.

Select the Notices as previously described in 5.4.6.1, “Distribute the File Package for MQSeries NT” on page 186:



Figure 170. Notice Group Messages (Courier)

Also the commqm.bat file invoked a MQSeries command to create the system and default objects. The output of this command is redirected in the file C:\DEFOBJ.OUT on the target machine. Therefore, you can view the output of the MQSC command by looking at the file on the target machine (in our case, WTR05121):

```
Starting MQSeries Commands.

AMQ8006: MQSeries queue created.
AMQ8010: MQSeries process created.
AMQ8014: MQSeries channel created.
AMQ8006: MQSeries queue created.
21 MQSC commands read.
0 commands have a syntax error.
0 commands cannot be processed.
```

5.4.6.3 Remove the File Package for MQSeries NT

After a file package has been distributed on a target, it can be removed using the Remove function. The remove operation will remove any files that were previously distributed and also trigger any Before or After distribution program.

Note that Courier does not perform any check if the package was previously distributed on the target since it does not have any *history* associated with the file package.

1. From the SoftDist policy region double-click on the **SD-NT** profile manager.

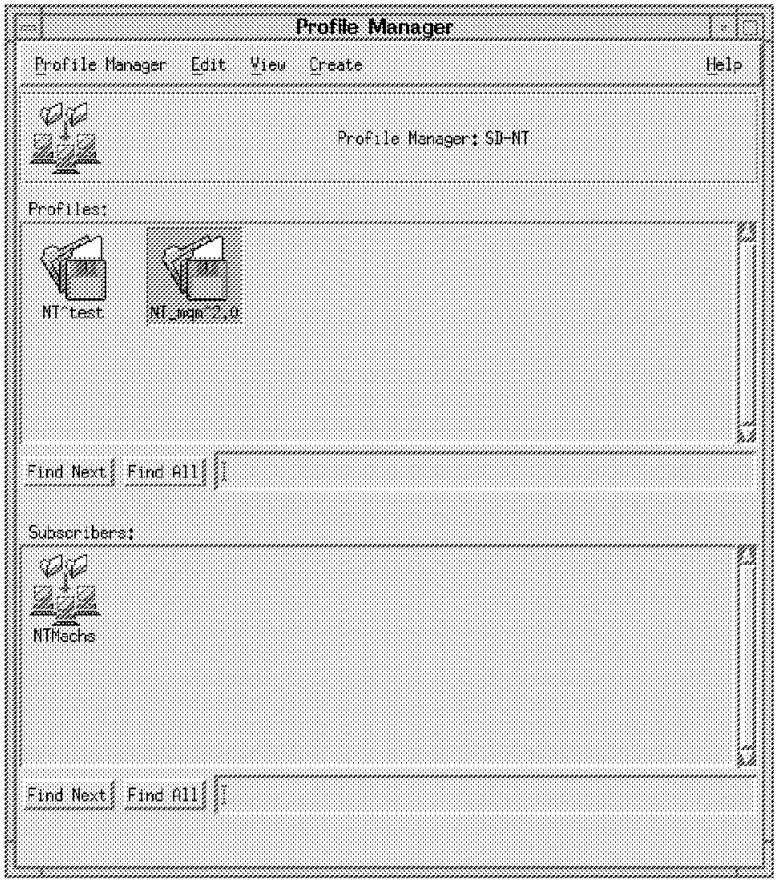


Figure 171. Profile Manager (SD-NT)

2. Double-click on the **NT_mqm^2.0** file package icon to display the windows:

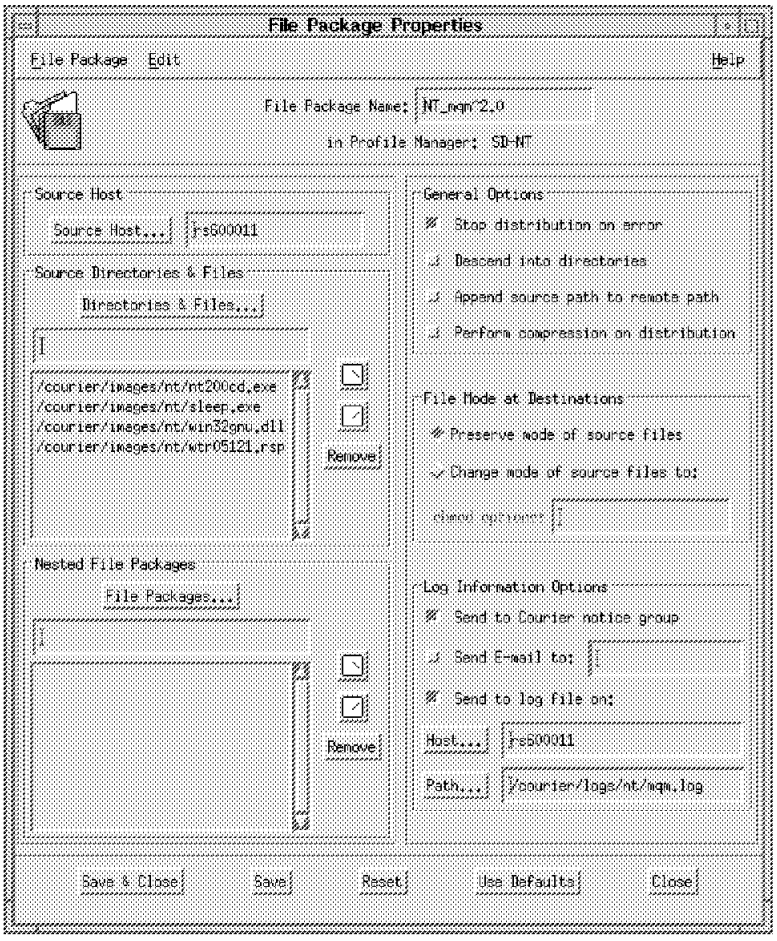


Figure 172. File Package Properties (MQSeries NT)

3. Select **File Package = > Remove from Hosts...** to display the following panel:

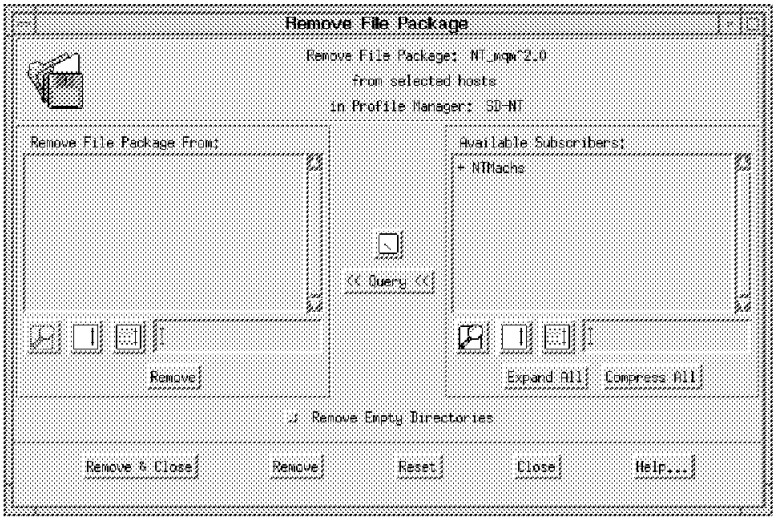


Figure 173. Remove File Package (MQSeries NT)

4. Fill in the Remove File Package From scrolling list with the subscribers from which you want to remove the file package.

You can choose the subscribers from the Available Subscribers list and move them to the Remove File Package From list using the arrow button.

In our case we have NTMachs as the profile manager in the Available Subscribers list. Therefore, we double-click on the profile manager, which is prefixed by the character + to display all of the managed nodes that subscribe to the profile manager. We then select **WTR05121**.

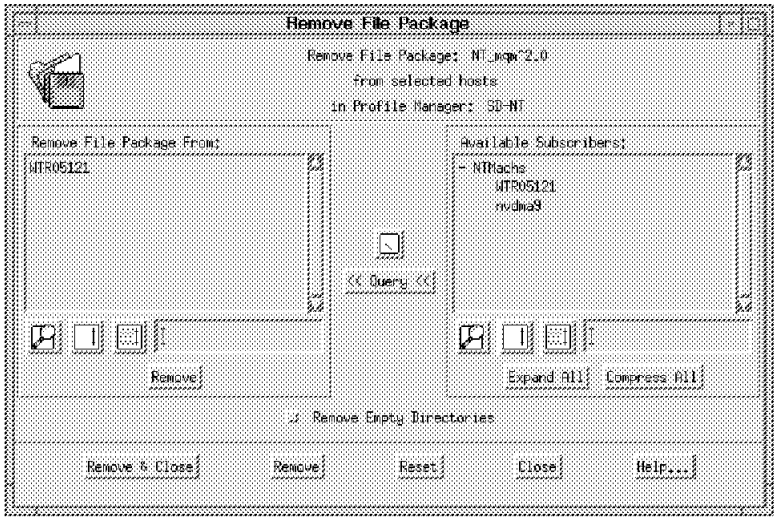


Figure 174. Remove File Package (MQSeries NT from Subscriber)

5. Select the **Remove & Close** button to remove the file package from the subscriber WTR05121.

 Note: The dialog will not be dismissed until the remove is complete. If the remove fails for any of the subscribers, a pop-up dialog is displayed to inform you on which subscribers the remove failed.

 Now Courier will start the Upon Removal script specified in the file package and then will remove from the target any source directories and files previously distributed.

 Remove Options

 Using the dialog you can only specify the Upon Removal script, which will be invoked only before the removal operation. If you want to run an after removal script, you need to export the file package, modify it, and then import it again. See 5.1.5, “Import/Export File Packages” on page 106 for more details.

Once the removal is complete, you can check the operation result by looking at the primary desktop panel, which will list the software distribution activities in the Operation Status field.

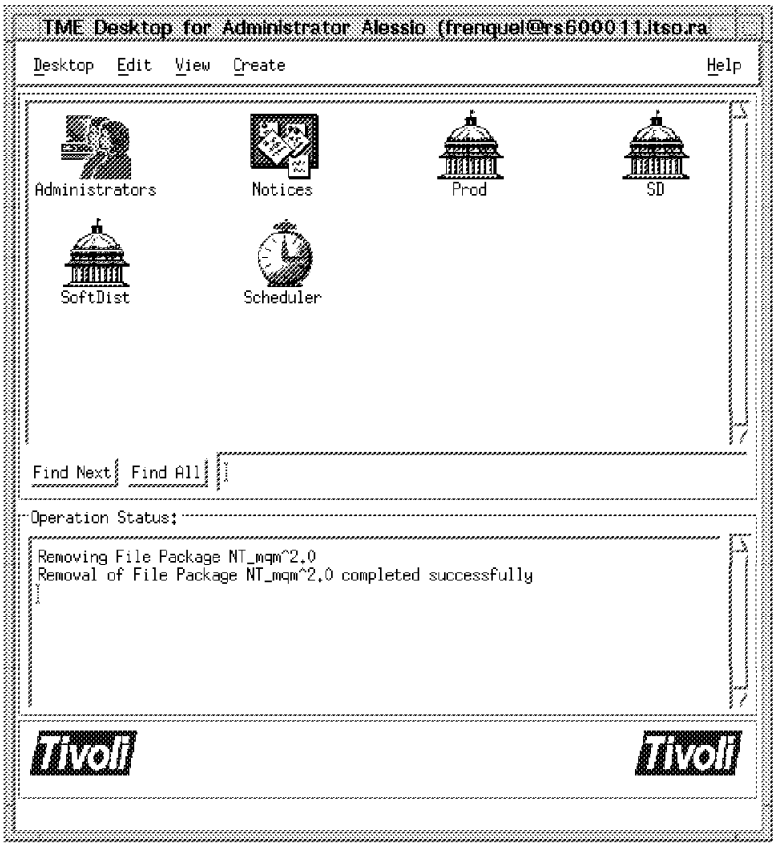


Figure 175. Desktop with Operation Status Messages

You also need to check the log file that we specified in the Log Information Options when we created the file package properties in 5.4.5.3, “Create the File Package Properties” on page 174.

The file /courier/logs/nt/mqm.log on rs600011 will contain the following messages:

```
================
File Package:  "NT_mqm^2.0"
Operation:     uninstall (m=1)
Finished:      Mon Aug 19 17:52:08 1996
--------
Source messages:
<none>
--------
WTR05121: SUCCESS
temp script: nt_after: C:\TEMP\nt_a2.bat
temp script: nt_commit: C:\TEMP\nt_c3.bat
temp script: nt_removal: C:\TEMP\nt_r4.bat
starting script: C:\TEMP\nt_r4.bat
'C:\TEMP\nt_r4.bat' script complete: status=0
uninstall: C:\nt200cd.exe: Not found
```

Note: The log reports c:\nt200cd.exe: Not found because Courier tries to remove the file previously distributed. The file was already deleted by our Commit program to gain space. Therefore, we can ignore the message.

We have also specified in the File Package Properties panel the Send to Courier notice group option, so that the Courier notice group will include notice messages for file package operations.

Select the Notices as previously described in 5.4.6.1, “Distribute the File Package for MQSeries NT” on page 186:

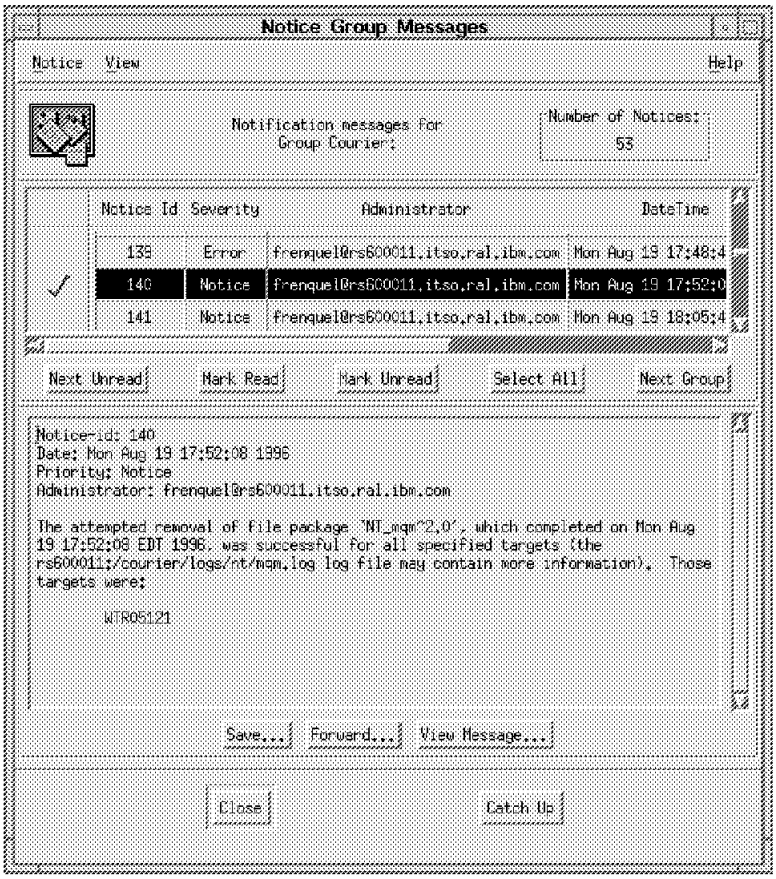


Figure 176. Notice Group Messages (Courier)

If you observe the NT server panel, the Uninstall program will be invoked and the MQSeries icon will be removed from the panel. Then once the uninstall is completed, it will reboot as instructed by Courier with the **Reboot machine After Removal** button set in 5.4.5.4, “Create the File Package Platform-Specific Options” on page 177.

5.5 Problems Encountered

In this section, we describe some of the problems we observed and what we did to correct these problems.

5.5.1 During Installation of Courier

Courier 3.0 requires Tivoli Management Platform (TMP) 3.0.1 installed on the TMR server or the managed node. Courier installation process will not check the version of TMP installed. If you attempt to install Courier 3.0 on a machine where you have only TMP 3.0 installed, the product installed dialog will display some error messages as shown in Figure 177 on page 203.



Figure 177. Product Install Dialog with Error Messages

To recover from this, you need to restore the system from a backup tape and install the service pack for TMP before you install Courier 3.0, or re-install TMP 3.0 from scratch, apply the service pack for TMP then install Courier 3.0.

5.5.2 Trying to Move a Managed Node from One TMR to Another

We wanted to move managed nodes from TMR Prod to TMR SD but without success. We had to delete the managed node from TMR Prod and remove the TME product from the machine.

Then we had to reinstall the TME product on the machine again, and connecting as an administrator in the TMR SD, we had to create the managed nodes again.

Chapter 6. Tivoli/Inventory

Tivoli Inventory Management provides information about hardware and software inventory on PCs and UNIX systems using the TME configuration repository and the TME profile structure; this information is stored in a central database. A TME administrator has the ability to view the inventory information for any node within the TME framework using the Tivoli Desktop GUI or command line interface (CLI).

Before we begin to install and customize the inventory application, we discuss the inventory terminology used in this section:

Scanning The process performed by the agent code to retrieve the inventory information and send this information to the database server.

MIF The inventory stores the information using the Desktop Management Interface (DMI) in the Management Information Format (MIF). This is an ASCII file that can be accessed using other utilities.

Inventory Profiles These are the profiles defined within the TME framework.

Signatures The software signatures identify the application package details on the client machines.

The Tivoli/Courier and Tivoli/Inventory integration allows the query functions to use information about each node to select the appropriate endpoints from a list of subscribers during a Tivoli/Courier distribution. In managing a large network with different machine configurations, the query function is helpful in using the information about which nodes meet prerequisite requirements for software distribution.

Administrators can use Tivoli/Inventory along with TME tasks to track unauthorized changes in hardware or software configuration. All of these tasks can be handled from one central management desktop.

See Chapter 7, “Tivoli/Courier and Tivoli/Inventory Interoperability” on page 249 for an example of how to gain full advantage of Tivoli/Inventory by combining it with Tivoli/Courier.

The Tivoli/Inventory examples covered in this section include the following platforms:

- AIX 3.2.5 and 4.1
- HP-UX 9.05
- SunOS 5.3
- MS Windows 3.11, Windows NT and Windows 95

At the time this book was written, the inventory product had the following restrictions:

- No OS/2 scanning support
- Only hardware inventory support for UNIX managed nodes (very limited)

© Copyright IBM Corp. 1996

For these reasons, we have concentrated on implementing Tivoli/Inventory for the PC managed nodes, with little discussion relating to the UNIX and NetWare servers. For an up-to date list of the supported platforms, see the *Tivoli/Inventory User's Guide* and *Tivoli/Inventory Release Notes*.

6.1 Tivoli/Inventory Components

The Tivoli/Inventory application is divided up into the following five main components:

- The inventory server component which resides on the TME server
- The PC inventory scanning component which must be installed on every PC for scanning purposes
- The UNIX inventory scanning component which must be installed on UNIX workstations for scanning purposes
- The control file data store component for managing PC system configuration
- The TME configuration repository component used with the TME application programming interface to define tables for storing inventory data on a database server

6.1.1 Tivoli/Inventory Server Component

The server component consists of a number of profiles controlled by profile managers. A profile contains all instructions to scan inventory from the managed nodes.

When the inventory profile is created by the administrator, the instructions contained in the profile determine the behavior of the scanning software, for example, the file name extensions, the file names searched for, and whether to re-scan the inventory or just retrieve a previous copy of the inventory Management Information Files (MIFs).

6.1.2 PC Inventory Scanning Component

The PC inventory scanning component is used to scan PC workstations to discover hardware, software and configuration files. This component describes the client part of inventory management software and must be installed on every PC that will be scanned for its inventory.

6.1.3 UNIX Inventory Scanning Component

The UNIX inventory component resides on a UNIX workstation to discover the hardware and software environment of the workstation. The inventory data is sent back to the server component to be stored in a database. This component describes the client part of inventory management software and must be installed on every UNIX-based node that is scanned for inventory data.

6.1.4 Control File Data Store Component

When Tivoli/Inventory performs a scan, files are stored in a repository in the TME database directory on the managing node where the PC managed node is connected. The files reside in a version control system and can serve as backup copies.

6.1.5 TME Configuration Repository Component

This component resides on the database server and defines the tables and fields where the software and hardware information is stored. The TME server communicates with the database module that acts as interface between RDBMS and the rest of the Tivoli/Inventory components.

6.2 Installation

The prerequisites for installing the Tivoli/Inventory application are described below:

- The Tivoli Management Platform (TMP) must be installed and running.
- All PCs that will be scanned require a supported TCP/IP stack.
- The TME PC Agent must be installed on each of the PCs.
- A relational database management system (RDBMS) must be installed on a server within the TME. The databases currently supported are ORACLE and SYBASE.
- TME managed nodes for each UNIX system and PC managed nodes for each PC system should be created.

In our project, we installed ORACLE 7.1.6 as our database. We recommend about 360 MB of DASD for the core database plus an additional 500 KB for each node's inventory data.

6.2.1 Installing the RDBMS

The relational database management system (RDBMS) should be installed before installing the Tivoli/Inventory application. You are asked to enter database environment variables during the Tivoli/Inventory installation.

If there is already a supported database system on your server, contact your database administrator for the required parameters, as they might be different from those we have chosen.

The following steps are required to install Oracle. In our project, the TMR server was on a RS/6000 (rs600011).

1. Create dba and oper groups, by typing the following:

   ```
   # mkgroup '-A' dba
   # mkgroup '-A' oper
   ```

2. Create a user named oracle:

   ```
   # mkuser pgrp='dba' groups='dba' home='/home/oracle' oracle
   ```

3. Create an ORACLE_HOME directory. We decided to use /usr/local/oracle. This directory needs about 310 MB of disk space.

4. Change the owner and group of that directory to oracle and group dba. Change the permissions to rwxr-xr-x, as follows:

   ```
   # chown oracle /usr/local/oracle
   # chgrp dba /usr/local/oracle
   # chmod 755 /usr/local/oracle
   ```

5. Create a bin directory. We chose /usr/lbin. Type the following:

   ```
   # mkdir /usr/lbin
   ```

6. Set your environment variables by creating and executing the following shell script named db_setup.sh:

```
#!/bin/ksh
export ORACLE_OWNER=oracle
export ORACLE_HOME=/usr/local/oracle
export PATH=$ORACLE_HOME/bin:/usr/lbin:$PATH
export ORACLE_SID=sid1
```

7. Create a directory as mount point for CD-ROM. Create another directory with about 50 MB of disk space to be used by the Oracle installation script. We named the directory /oracle_link. See Figure 178.

```
$su root
# mkdir /mount_point
# chmod 777 /mount_point
# mkdir /oracle_link
# chmod 777 /oracle_link
# mount -rv cdrfs /dev/cd0 /mount_point
# exit
$
```

Figure 178. Mount the CD-ROM Drive

8. Provide links from UNIX filenames to the CD-ROM. This process may take up to ten minutes. Type the following:

```
# cd /mount_point/orainst
# ./start.sh
```

9. Install post-wait driver kernel extensions, as follows:

```
# cd /oracle_link/orainst
# ./rootpre.sh
# exit
$
```

10. Run the installer. This process may take one hour. Type the following:

```
$ id
uid=202(oracle) gid=203(dba) groups=1(staff)
$ cd /oracle_link/orainst
$ ./orainst
```

You are asked for the following responses when running the Oracle installation script:

a. Reenter the value of ORACLE_HOME as /usr/local/oracle.

b. Confirm that rootpre.sh has been run.

c. Reenter the value of ORACLE_OWNER as oracle.

d. Reenter the value of ORACLE_SID as sid1.

e. Select the products you want to install. We installed the following products:

- Oracle Data Query
- Oracle Easy*SQL
- ORACLE7 Server (RDBMS)
- Oracle UNIX Installer and Documentation

f. You are prompted to select privileged groups and to set passwords. The defaults worked for us.

11. Log in as root and set your environment by running `db_setup.sh`.
12. Run the post installation task, as follows:

    ```
    # cd $ORACLE_HOME/orainst
    # sh ./root.sh
    ```

 If a message that ORACLE_HOME is not the same as the home directory for this user appears, continue.
13. Edit `/etc/oratab` and change `sid1:/usr/local/oracle:Y` to `sid1:/usr/local/oracle:N` to bring up the database at system reboot time.
14. To automatically start the database at system startup, insert the following line:

    ```
    # mkitab "oracle:2:wait:/bin/su oracle -c $ORACLE_HOME/bin/dbstart"
    ```

6.2.2 Installing Tivoli/Inventory

Before installing Inventory, have Tivoli Management platform (TMP) installed on your system. See 2.4, "Installing the TME Management Platform" on page 23. To install Tivoli/Inventory Management, the administrator needs at least the install_product and the senior authorization roles set. We installed the product by using the GUI interface. In our project, the Tivoli/Inventory server (rs600011) was running AIX Version 3.2.5.

Starting Tivoli will open the administrator desktop window on your display. Select **Install Product** by moving the mouse pointer over the Desktop menu and pressing the left mouse button. Scroll down to Install and release the mouse button over Install Product.

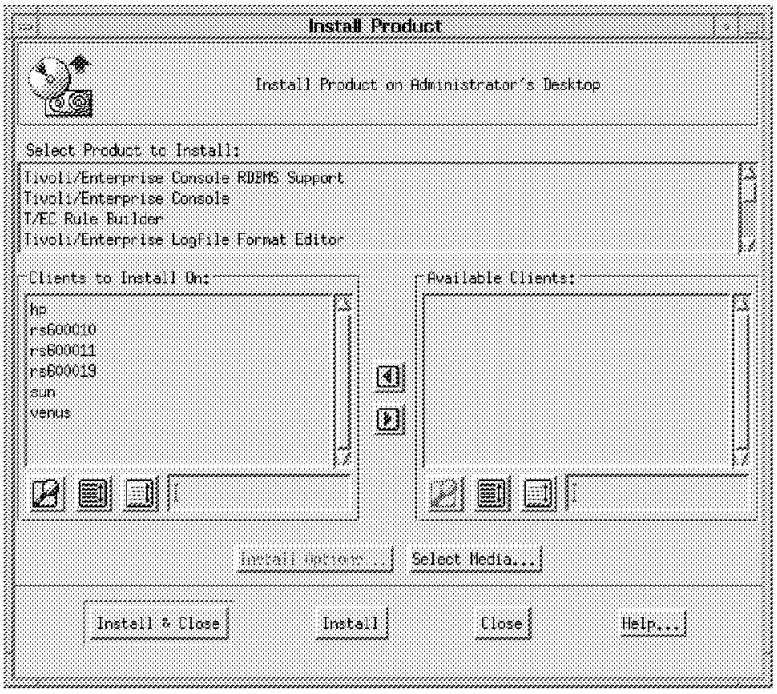


Figure 179. Install Product Window

Click on the **Select Media** button to enter the installation directory where the CD-ROM is mounted over.

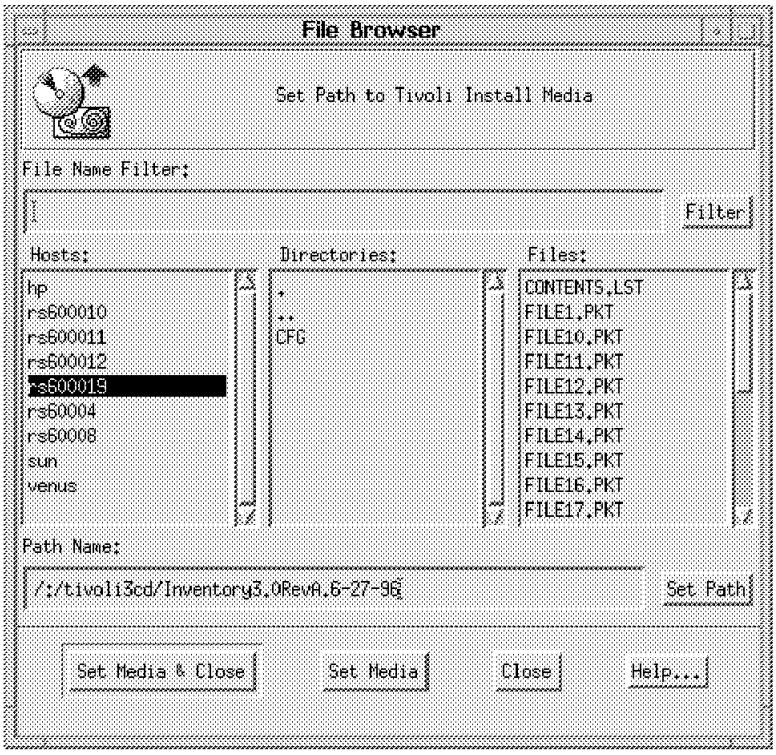


Figure 180. File Browser Window. Enter directory name for installation.

In our project we copied the necessary files to a directory of a file system called tivoli3cd. After inserting the correct directory path, close the window by selecting **Set Media & Close**.

NOTE

When Tivoli product files are to be copied to hard disk, the file names must be uppercase.

Select **Tivoli/Inventory Application Postscript Documentation** from the Install Product window. Insert the directory name where files should be installed on your server at the Install Options pop-up window and click on the **Set** radio button to close the window again.

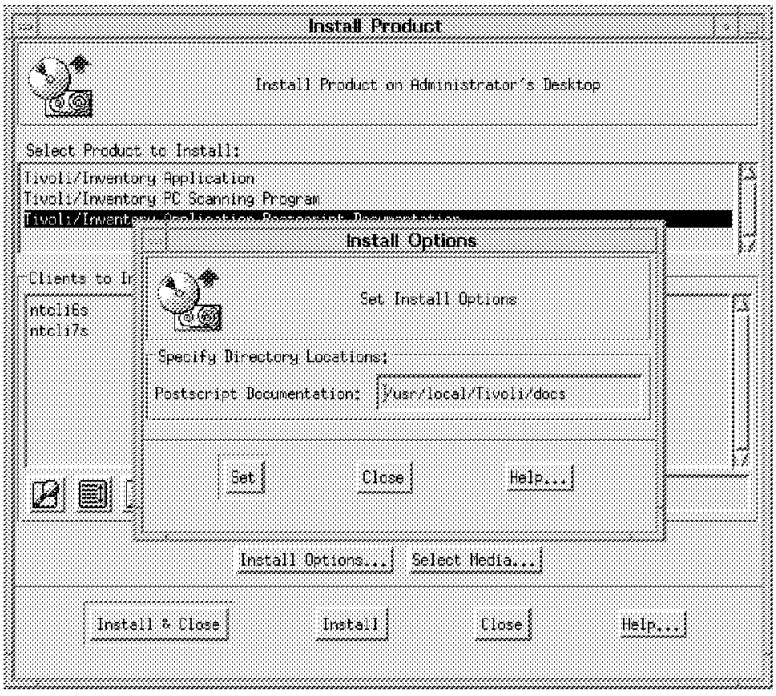


Figure 181. Install Options Window (Postscript Documentation)

The Tivoli/Inventory Application PostScript Documentation, /usr/local/Tivoli/docs/inv_user_guide.ps, contains the *Tivoli/Inventory User's Guide*, which you can print on a PostScript printer.

Select **Tivoli/Inventory Application** from the Install Product window to install the Tivoli/Inventory Application.

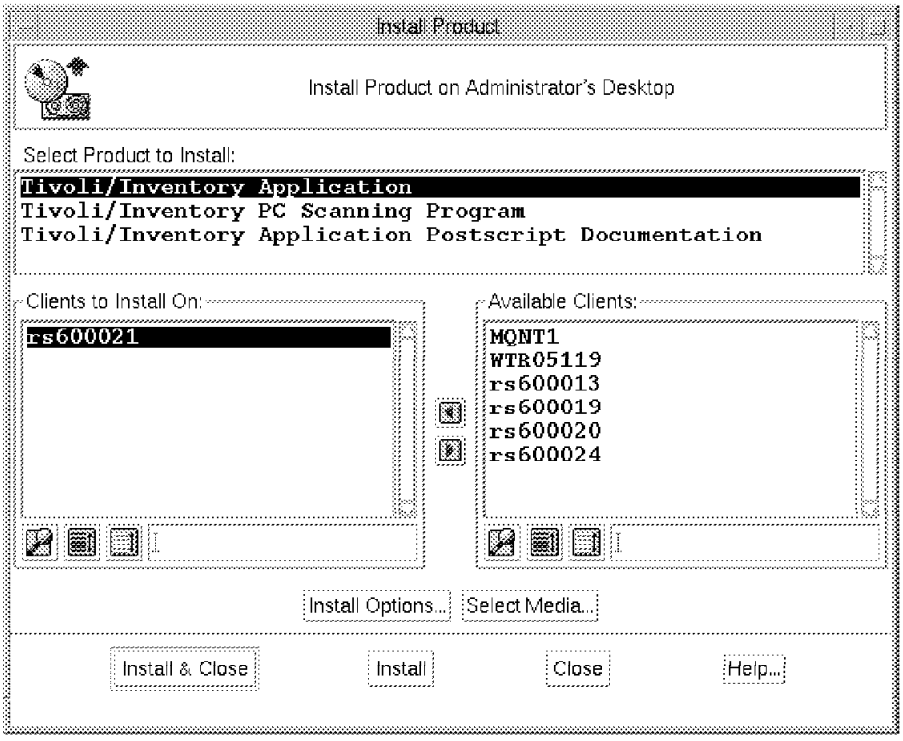


Figure 182. Install Product Window (Inventory Application)

The Install Options window prompts you for RDBMS specific information. See the *Tivoli/Inventory User's Guide* about the possible options. The Database ID is the same as the ORACLE_SID you specified when you installed the Oracle product. See 6.2.1, “Installing the RDBMS” on page 207 for this information. The Server ID field is blank for installation on the Tivoli/Inventory server. If we are installing Tivoli/Inventory on another managed node, enter your Tivoli/Inventory server name (rs600011) in the Server ID field. See 6.2.6, “Installing Managed Nodes” on page 216 and Figure 188 on page 217 for information on installing managed nodes.

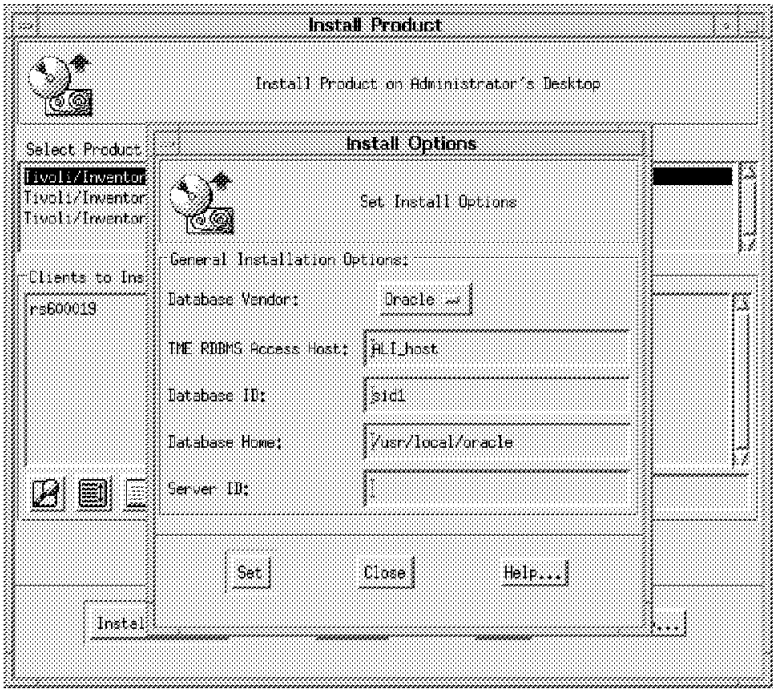


Figure 183. RDBMS-Specific Data

Select the **Install** push button to start the installation. Follow the instructions in the Product Install window.

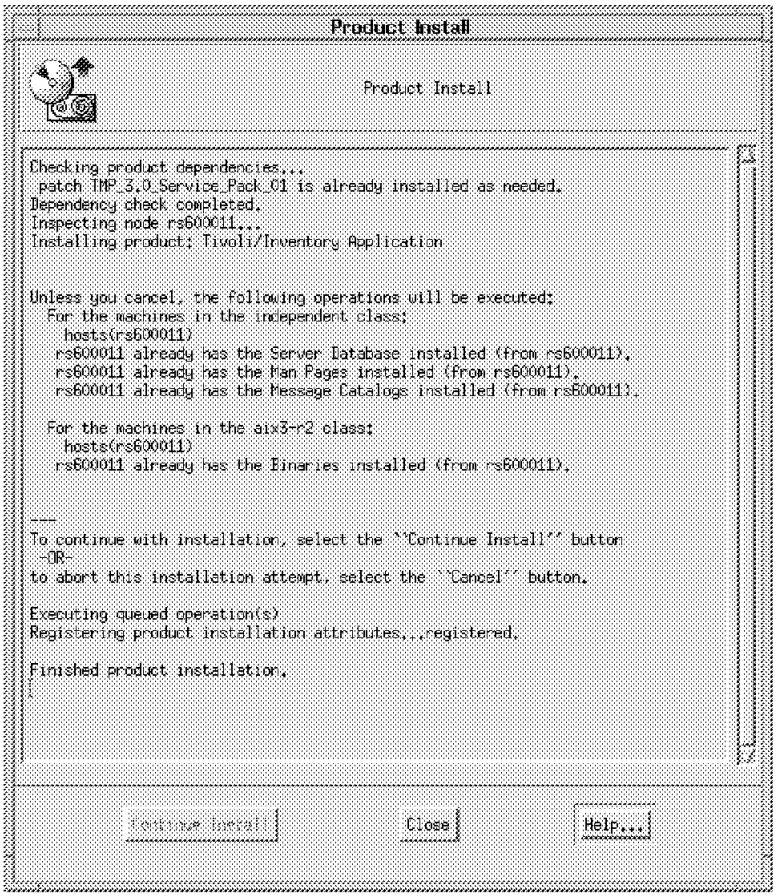


Figure 184. Product Install Window

6.2.3 Creating the Inventory User Account on the RDBMS

To create the necessary tables in RDBMS required by Tivoli/Inventory, run two shell scripts, which are provided by the product and are located in the /usr/local/Tivoli/bin/aix3-r2/TAS/RIM/SQL/scripts directory.

To execute the scripts, log on or switch to AIX *oracle* user. To execute the SQL scripts, you can log on to the database as an Oracle *sys* user.

As the AIX oracle user, type the following:

```
# cd /usr/local/Tivoli/bin/aix3-r2/TAS/RIM/SQL/scripts
# sqlplus
```

Oracle prompts you to enter the user name. Enter sys and the password. You specified the sys password when you installed the Oracle product.

You should be logged on to the oracle database. To execute the first SQL script, enter the following command:

```
SQL> @tivoli_ora_admin.sql
```

The script will create the RDBMS tables for Oracle and add a new Oracle user called *tivoli*. Log out as an Oracle user *sys* by entering `quit` at the Oracle command line.

6.2.4 Installing Configuration Repository Database

Log on to the RDBMS as Oracle user tivoli with the password of tivoli. Run the second script by entering the following command at the oracle command line:

```
SQL> @tivoli_ora_schema.sql
```

When you run the script for the first time, some error messages will occur because Tivoli script drops tables from the database that have not been created yet.

You should change your *tivoli* password. The password can be changed by issuing the `wsetrimpw inventory` command.

Do not change the user name because Tivoli/Inventory relies on this information to communicate with the RDBMS.

To verify that your database is set up correctly, we logged on to the database by entering `sqlplus`. Choose user ID sys and the password we specified at database installation time. Enter `select * from all_catalog where owner = 'TIVOLI';` to show all tables that are used by Tivoli products.

6.2.5 Installing the PC Managed Nodes

For every PC managed node that is to be scanned for inventory, we had to install the PC portion of the scanning program. To install the PC scanning program from the TMR server, the TME PC Agent program must be installed on the PC managed nodes.

We installed the TME PC agent Version 3.0.1 on our PC managed nodes. TME PC agent Version 3.0.1 must be used on Windows 3.11 and Windows 95 PC managed nodes to prevent a general protection fault error during installation. Refer to 2.4.5, “Installing PC Managed Node as TME Client” on page 31 for detailed information on how to install the TME PC Agent program.

To install, select **Tivoli/Inventory PC Scanning Program** from the Install Product window. You can set the directory (where the software is installed on the agent) in the Binaries field on the Install Options dialog. Select on **Set** button to close the Install Options window.

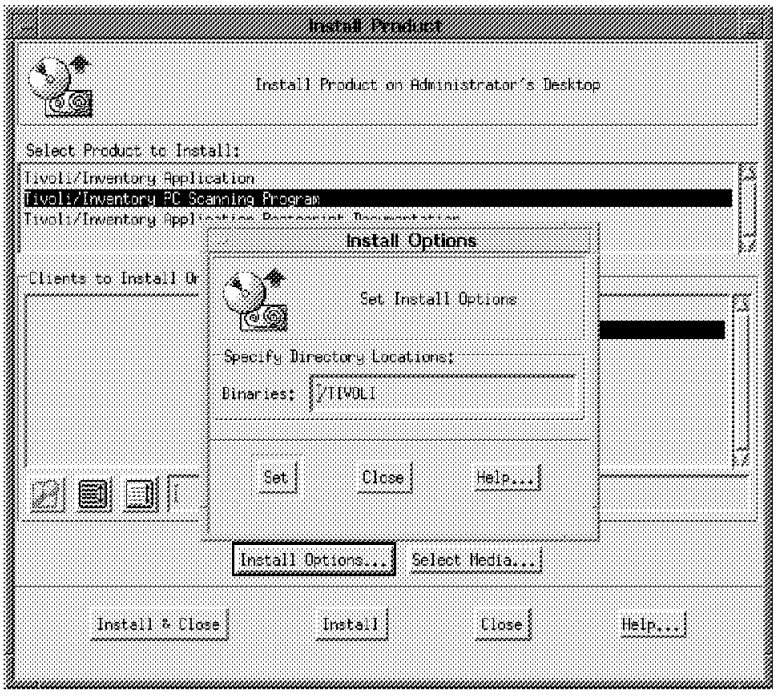


Figure 185. Install Options Window (PC Scanning Program)

The list of available clients will show you all known PC managed nodes. For a selection of clients to be installed, select one or more from the list and select the left arrow radio button. The nodes will disappear from the right side and show up on the left side.

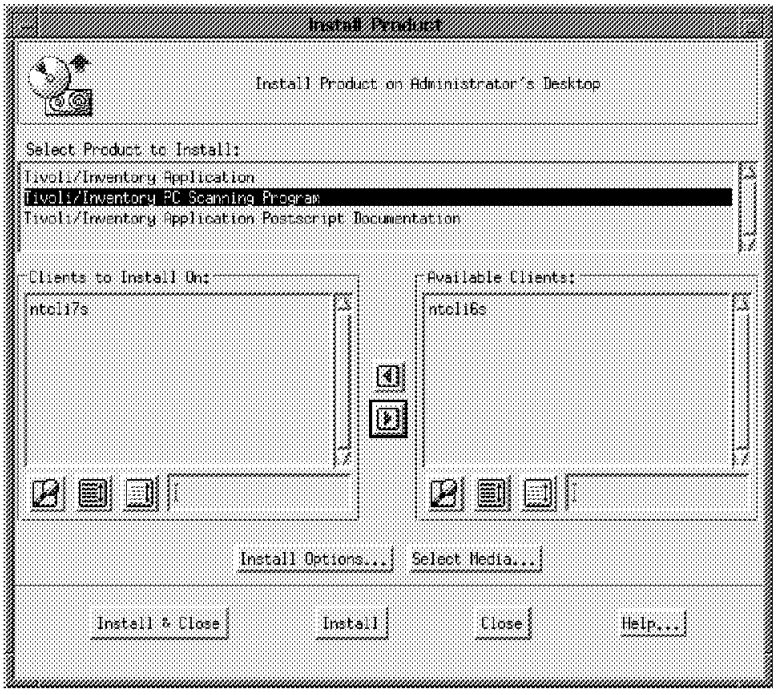


Figure 186. Install Product Window (PC Scanning Program)

Clicking on the Install button will bring up the **Install Product** window, which gives information about the status of the procedure.

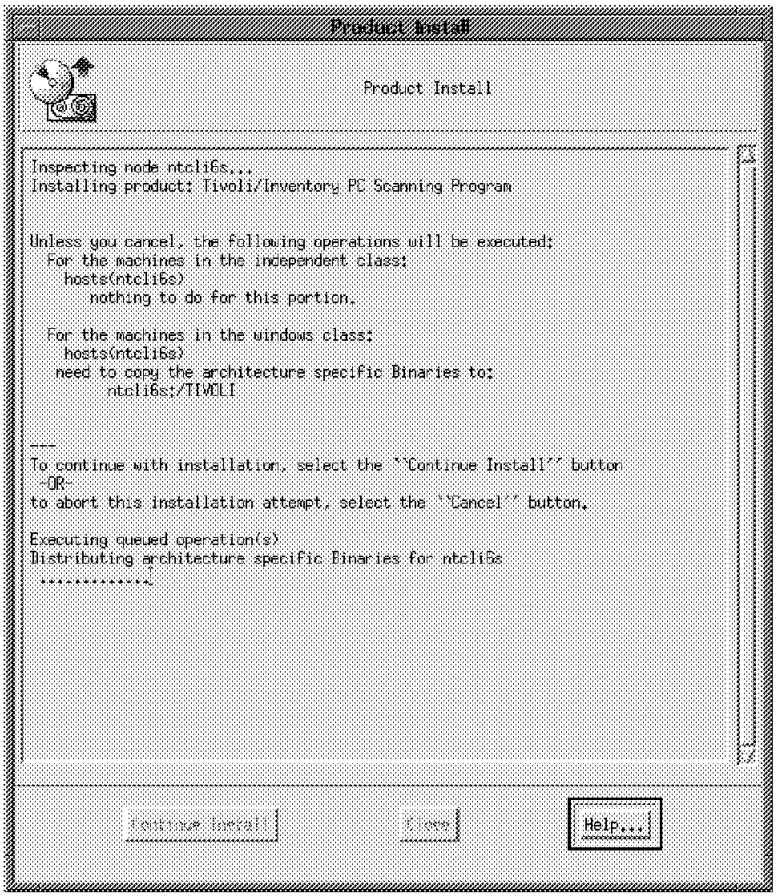


Figure 187. Product Install Window (PC Scanning Program)

6.2.6 Installing Managed Nodes

Every UNIX system that is scanned for inventory must have the Tivoli/Inventory Application installed. To install the Tivoli/Inventory Application, the Tivoli Management Platform must be installed. For installation of Tivoli/Inventory, refer to 6.2.2, “Installing Tivoli/Inventory” on page 209. If we are installing Tivoli/Inventory on another managed node, enter your Tivoli/Inventory server name (rs600011) in the Server ID field. See Figure 188 on page 217.

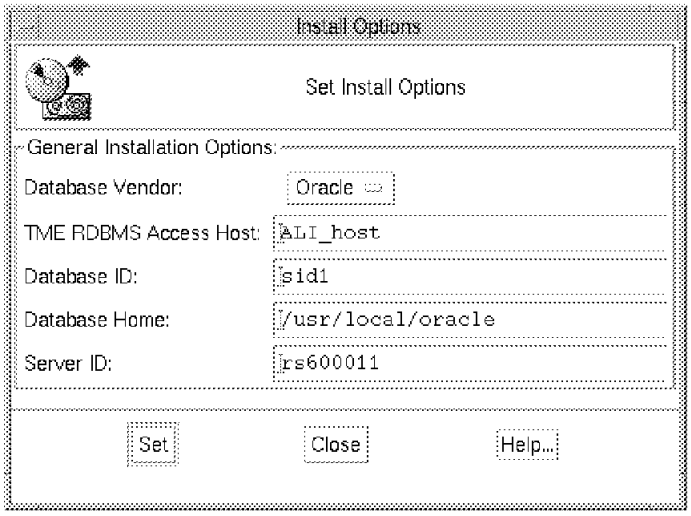


Figure 188. Install Options Window (Inventory on TME Client)

6.3 Using Tivoli/Inventory

This section describes how to set up an inventory profile to get information from different workstations and PCs in a heterogeneous network. You will see what tasks have to be done to get inventory data, how you will be able to view the data and how the scanning process works on both platforms (PC managed nodes and managed nodes).

6.3.1 Set Resource Roles for Inventory

Figure 189 on page 218 shows how to set the Tivoli/Inventory resource roles for Administrator.

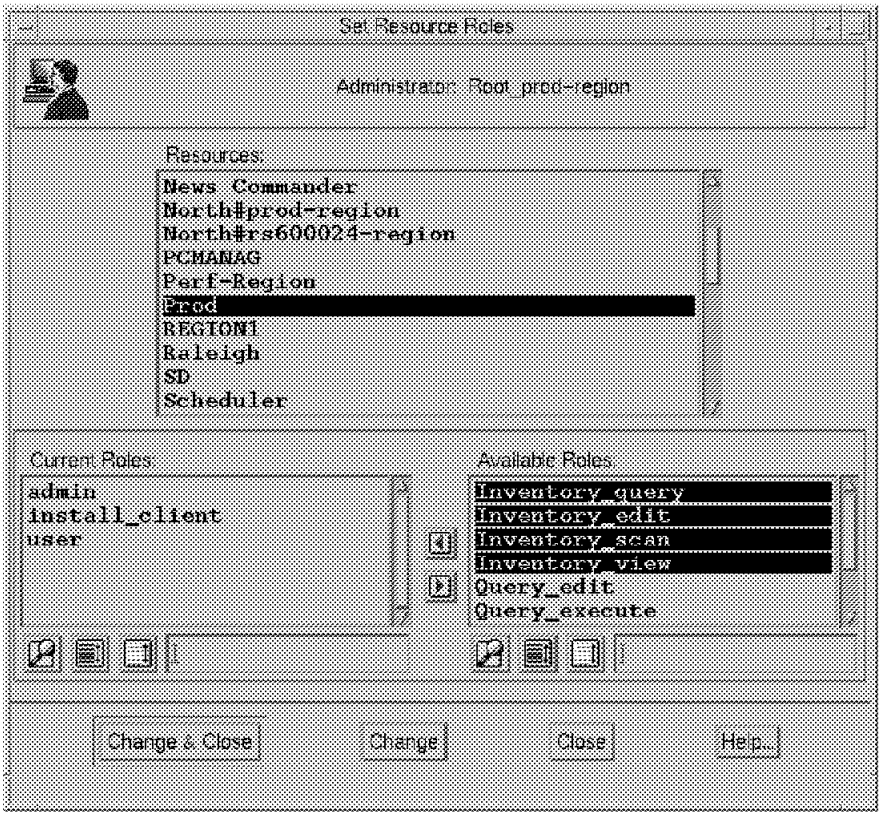


Figure 189. Set Resource Roles (Inventory)

6.3.2 Set Up an Inventory Profile

Tivoli/Inventory is a profile-based TME application, and as such uses management by subscription. This means that the scanning instructions are defined in profiles that have been created under profile managers.

The profile manager is contained within a policy region. An inventory profile is contained within a profile manager. Managed nodes, PC managed nodes, and profile managers can subscribe to the profile manager. These subscribers can be managed by the application that uses this profile manager.

The steps that must be completed before you can create an inventory profile are listed below:

1. Create a policy region
2. Create a profile manager
3. Subscribe the manage nodes to the profile manager

For a detailed description on how to create a policy region, refer to 3.1.2, “Create Policy Regions” on page 61. For a detailed description on how to create a profile manager, refer to 3.1.5, “Create Profile Managers” on page 68.

6.3.2.1 Creating an Inventory Profile

To create an inventory scanning profile, open the profile manager by double-clicking on the Profile Manager icon.

The profile manager we defined was called INVENTORY. Click on the **Create** button from the top menu bar, enter the name of your profile and select the **Create & Close** button. The profile we created was named INVSCAN1.

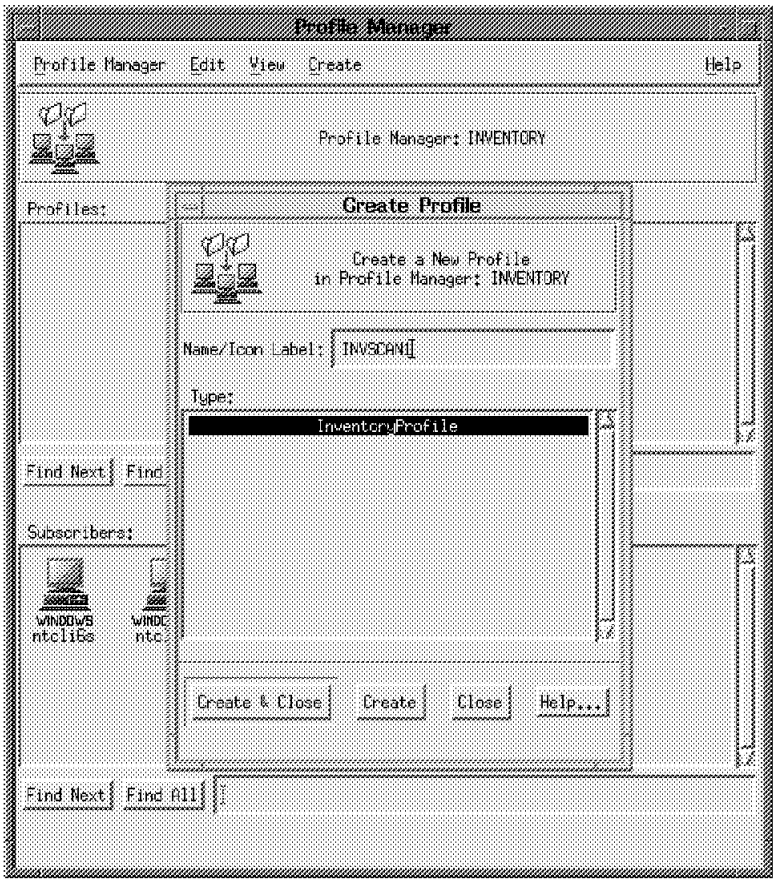


Figure 190. Creating a Profile

After the profile is added to your profile manager, you will see a window as shown in Figure 191 on page 220.

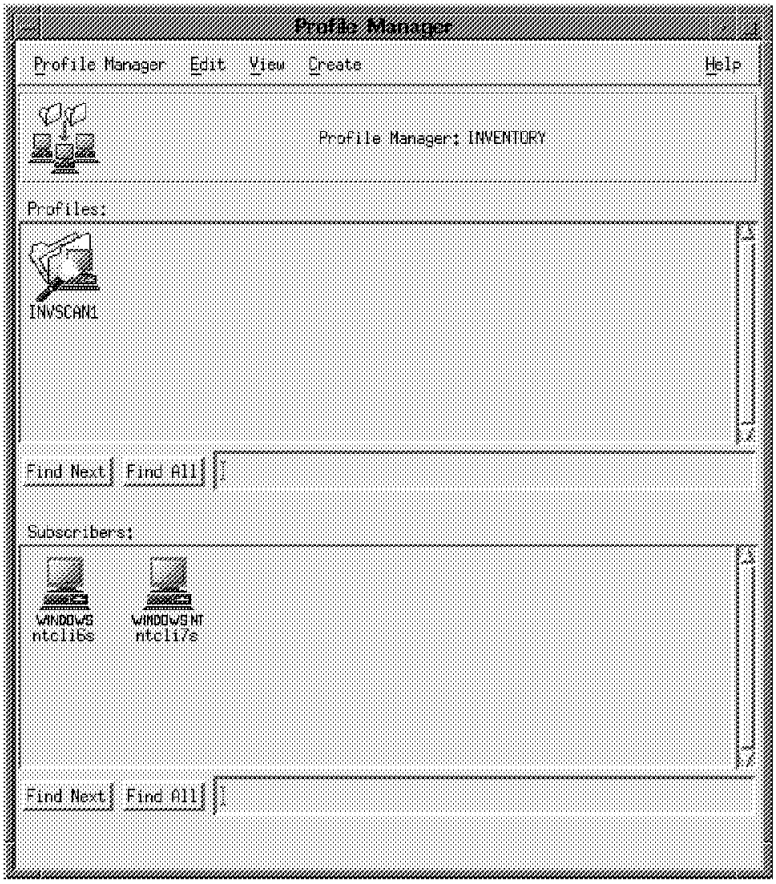


Figure 191. Profile Manager Window (with Profile INVSCAN1)

6.3.2.2 Customizing Inventory Profile

Customizing an inventory profile allows you to change the scanning instructions and requirements. You can use the default policies or reset them to new values. You are able to add software signatures to track any specific application not currently supported by the Tivoli/Inventory default lists. To add special signatures, open the Inventory Profile by double-clicking on the profile icon.

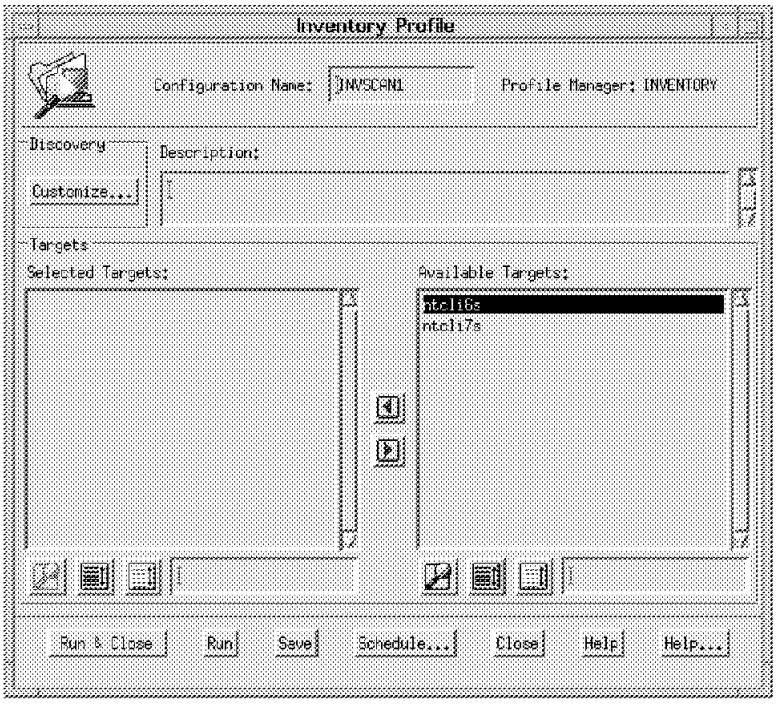


Figure 192. Inventory Profile Window

Select the **Customize** push button to open the Customize Inventory Retrieval window.

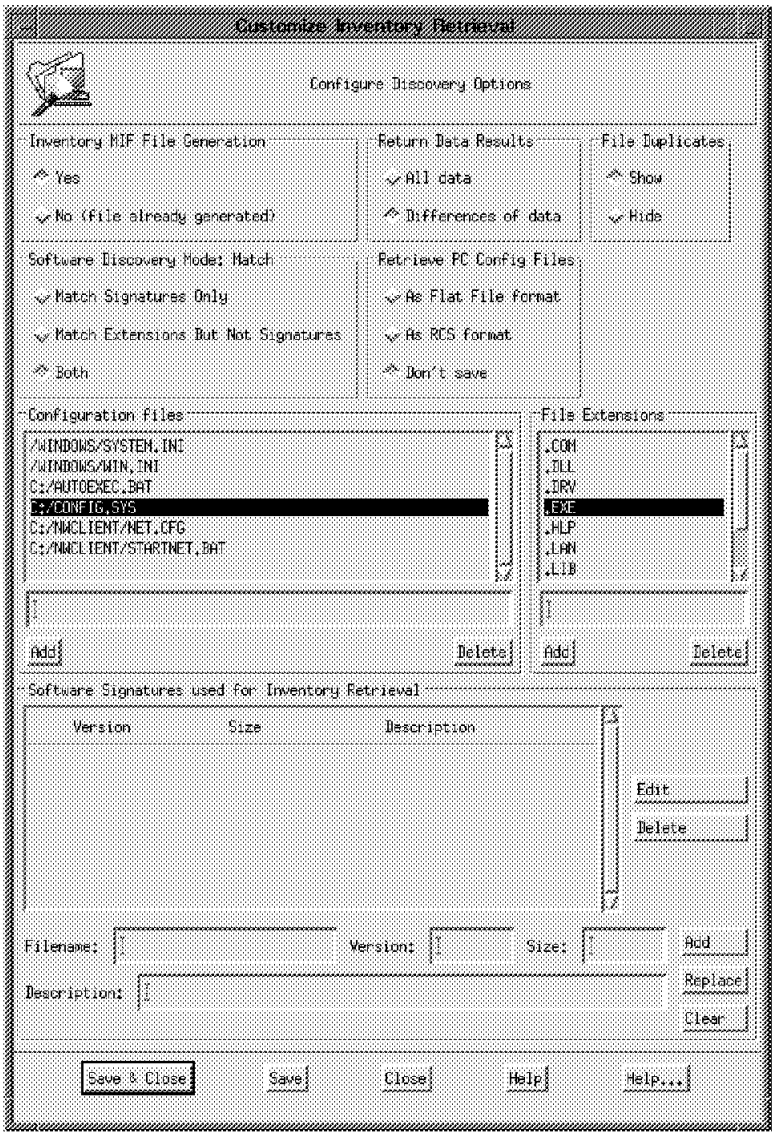


Figure 193. Customize Inventory Retrieval Window

You can change the behavior of the scanning program by selecting options and by entering your values in the appropriate options.

NOTE

Currently, this function is only supported on PC managed nodes.

For a description of the options, refer to the *Tivoli/Inventory User's Guide*.

- Inventory MIF File Generation

 Specifies if a MIF file should be generated at the target, or if a MIF file already exists on the target.

- Return Data Result

 Specifies whether Tivoli/Inventory should return all inventory information or compare the most recent results of a scan with the current contents of the

configuration repository. For initial scanning, this option should be set to All data.

- File Duplicate

 Specifies whether to record existing files that are duplicated in more than one directory, or ignore the duplicates.

- Software Discovery Mode: Match

 Specifies whether to search for files that match the software signatures only, or to track files that match the extensions in the file extensions list, or both.

- Retrieve PC Configuration Files

 Specifies if the PC configuration file should be retrieved and stored as a flat file, or as a revision control system (RCS) formatted file. These files are be stored in the /var/spool/Tivoli/<server-name>.db/file_versions directory on the Tivoli/Inventory server. Refer to the Tivoli Management Platform Planning and Installation Guide for more information about RCS.

- Configuration Files

 Specifies the list of PC configuration files to be searched for at the PC.

- File Extensions

 Specifies file extensions to be searched for at the PC.

- Software Signatures used for Inventory Retrieval

 Specifies software signatures for any specific applications you would like to track, but are currently not supported by Tivoli/Inventory.

 The file name must be in capital letters. You may not be able to discover files without an extension. You should specify a file extension to the file name and your file extension must be specified in the file extensions window. The size must be specified as at least three digits, that is, for 28 bytes, specify 028 instead of 28.

6.3.3 Scanning the Environment

After creating and customizing an inventory profile, you can gather the information from PCs and UNIX systems.

When Tivoli/Inventory gathers information, it starts one type of scanning software on UNIX platforms and another on PCs. For detailed information on how the scanning process works, see 6.3.3.2, “The Scanning Process on PC Managed Nodes” on page 228 and 6.3.3.3, “The Scanning Process on the UNIX Managed Nodes” on page 231.

The first time Tivoli/Inventory activates the scanning on each subscribing node:

- The scanning software reads inventory information from that system.
- The scanned information is sent to the TME server.
- The scanned information is stored in the configuration repository.

This initial scan could take a while in a large enterprise.

For subsequent scanning operations, you can set the scanning instructions in the profile to have the scanning software only compare the result of the current scan with the result of the previous scan. This option causes Tivoli/Inventory to send

only the differences to the configuration repository. The changes will be merged with the results of previous inventory gathering operations.

After Tivoli/Inventory has scanned subscriber systems, it creates a MIF file containing all the inventory information from the scanning process. The MIF file is converted into a standard database format, consisting of a set of tables representing the inventory information. These tables can be extended or customized. Refer to the *Tivoli/Inventory User's Guide* for more information on this topic.

To gather inventory information from subscribers, the administrator must have the Inventory_scan, senior, or super roles set (see 3.1.7.1, "Creating Administrators Using the Graphical Interface" on page 72).

6.3.3.1 Activating a Scanning Process

To activate the scanning process, double-click on the inventory profile, This will display the Inventory Profile window. This window shows the available targets on the right and the selected targets on the left.

To move the nodes to the Selected Targets list box, select the node using the left mouse button and click on the left arrow radio button as shown in Figure 194.

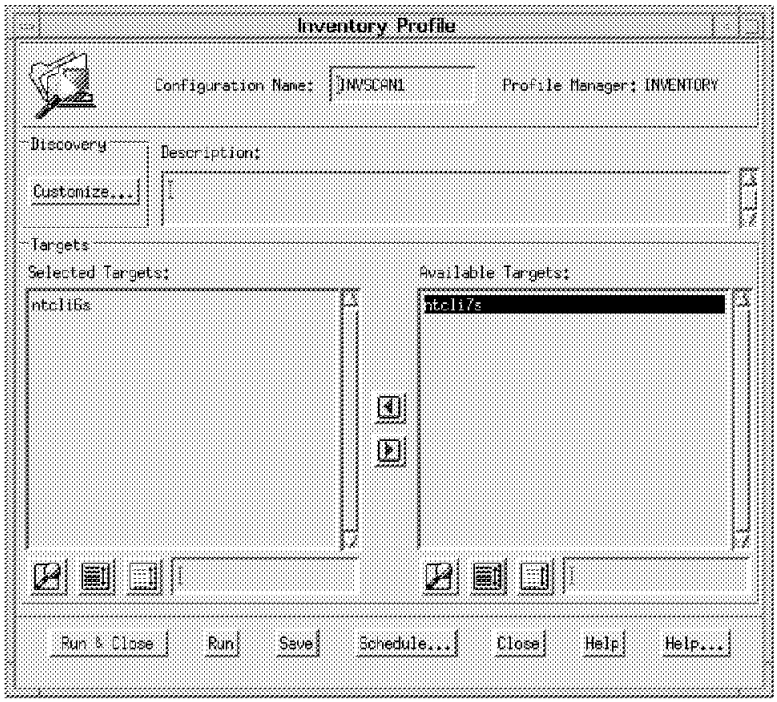


Figure 194. Selecting Targets for Inventory Scanning

The initial inventory scan select radio requires the All data option to be selected from the Customize Inventory Retrieval window. To start inventory scanning from the selected targets, click on the **Run & Close** button.

Another way to activate the scanning process on the clients is to select the profile by clicking and holding down the middle mouse button. The profile icon is dragged using the middle mouse button over the top of the subscriber icon and dropped on the subscriber icon. This operation will activate the scan for that particular subscriber. The inventory gathering process begins on this subscriber.

You can schedule an inventory gathering operation by selecting the **Schedule...** radio button, as shown in Figure 195 on page 225.

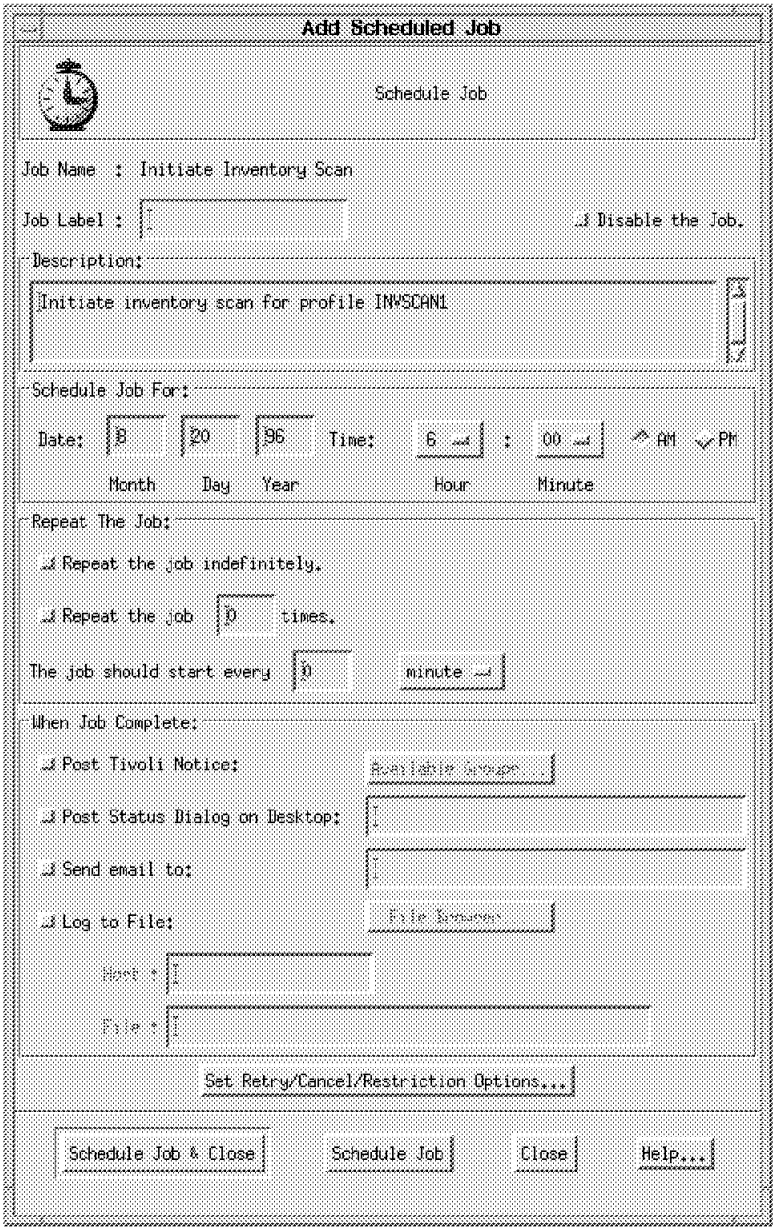


Figure 195. Add Scheduled Job

To get information about the status of scanning, select the **Notices** icon on your desktop. You can open the Read Notices window by clicking and holding down the right mouse button over the icon and releasing it with the mouse pointer over the Read Notices option on the pop-up menu.

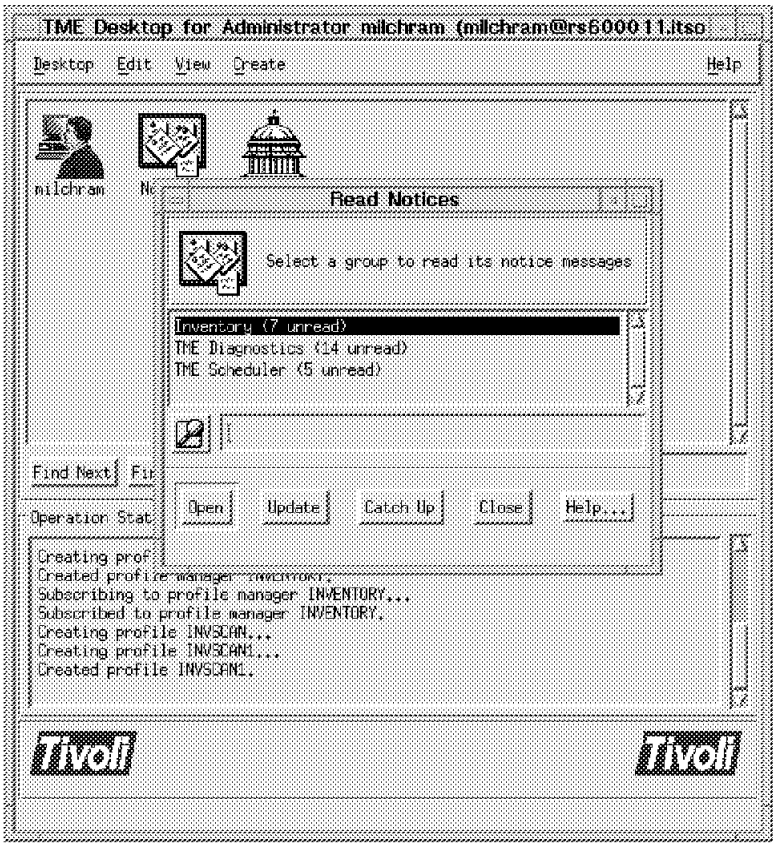


Figure 196. Read Notices Window

Select **Inventory** to get a display of notices belonging to an inventory scan. Select the **Open** button to view the Notice Group Messages window.

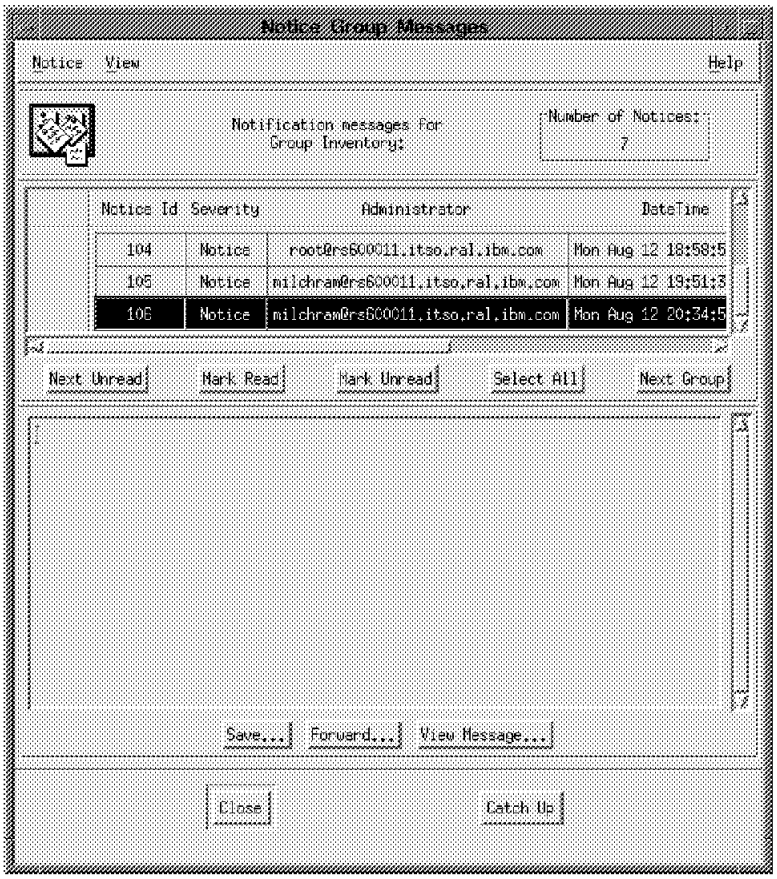


Figure 197. Notice Group Messages Window

To get detailed information, select a message from the list and click on the **View Message** radio button. The Notice Message Viewer window appears and gives you detailed information about the last distribution. The message you view will then be marked as Read. Select the **Catch Up** button to remove all messages from the window.

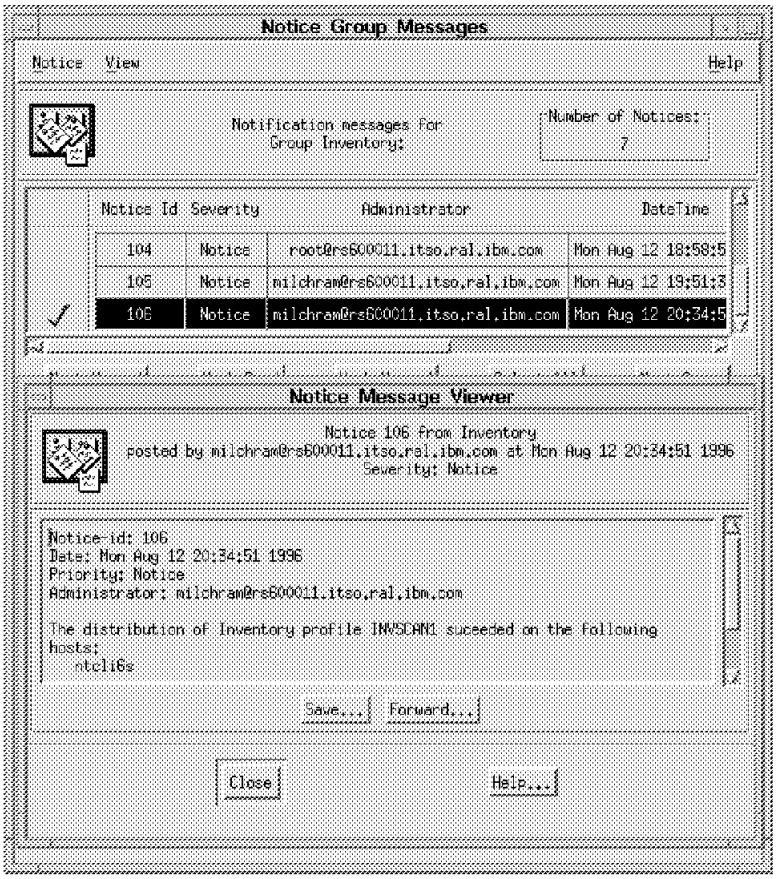


Figure 198. Notice Message Viewer

6.3.3.2 The Scanning Process on PC Managed Nodes

Inventory information (hardware, software, and configuration files) is discovered by the PC inventory scanning component.

For PC endpoints, the scanning is done using the Intel LANDesk scanner, which runs on Windows 3.x, Windows 95 and Windows NT. The Intel LANDesk scanner doesn't scan the Windows NT Registry for any information. A standard list of commercially available application signature files is provided by LANDesk in the file LDAPPL.INI. User-defined software signature file names can be added, but they are kept separate from the LDAPPL.INI applications.

The Tivoli/Inventory Application creates for each PC managed node a directory under the /var/spool/Tivoli/<server-name>.db/inventory directory on the server, where server-name is the name of the Tivoli/Inventory server (rs600011).

You can see the PcManagedNode resources defined within the TME object database by issuing the command:

```
wlookup -r PcManagedNode -a
```

Two files are created at the server and sent to the PC managed node and executed there. The first file called TIVSCAN is a script (see Figure 199 on page 229).

```
cd \TIVOLI\SCANManagedNode -a
del .\OUTPUT\LDISCAN.MIF
del .\OUTPUT\OUTFILE.TXTlso used to store data returned by the agent.
del .\OUTPUT\MIFCHECK.SUM
del .\OUTPUT\OUTCHECK.SUM
.\mergeini .\INI\LDAPPL.INI .\INI\USERADD.INI .\LDAPPL.INI
.\ldiscan /o=.\OUTPUT\OUTFILE.TXT /m /u /w
.\checksum .\OUTPUT\LDISCAN.mif > .\OUTPUT\MIFCHECK.SUM
.\checksum .\OUTPUT\OUTFILE.TXT > .\OUTPUT\OUTCHECK.SUM
```

Figure 199. TIVSCAN

The second file called useradd.mif includes customization settings from the inventory profile (see Figure 200).

```
MIFPATH=\TIVOLI\SCAN\OUTPUT
ScanExtensions=.EXE .COM .SYS .NLM .DLL .DRV .LIB .HLP .LAN .ORI
&lbracket.Applications&rbracket.
Mode=All
Duplicate=ON
CfgFiles1=C:\CONFIG.SYS C:\NWCLIENT\STARTNET.BAT
CfgFiles2=C:\AUTOEXEC.BAT C:\NWCLIENT\NET.CFG
CfgFiles3=\WINDOWS\WIN.INI
CfgFiles4=\WINDOWS\SYSTEM.INI
```

Figure 200. The useradd.mif Script

At the PC managed node, the executable programs are installed at the directory \TIVOLI\SCAN.

The directory \TIVOLI\SCAN\INI includes the file LDAPPL.INI which consists of a list of all currently supported applications.

The MERGEINI program merges the data files, LDAPPL.INI and USERADD.INI, which were sent to the PC managed node, into one file called LDAPPL.INI that will be stored at \TIVOLI\SCAN\.

The scanning program, named LDISCAN, uses this file as input and creates two output files called outfile.txt and ldiscan.mif. These two files will be sent to the server and stored as outfile.txt and mif.last in the agent directory. If an older file mif.last exists, it will be renamed to mif.last.last. Tivoli is able to determine any differences between two scanning processes. The data structure stored in the MIF (Management Information File) file complies with the standard defined by DMTF (Desktop Management Task Force). The MIF file is sent to the server and parsed to put the data into RDBMS.

For our project, we added the file name TESTFILE.TXT with version, description and size as a user-defined software signature as shown in Figure 201 on page 230. For more details on customizing an inventory profile, see 6.3.2.2, “Customizing Inventory Profile” on page 220.

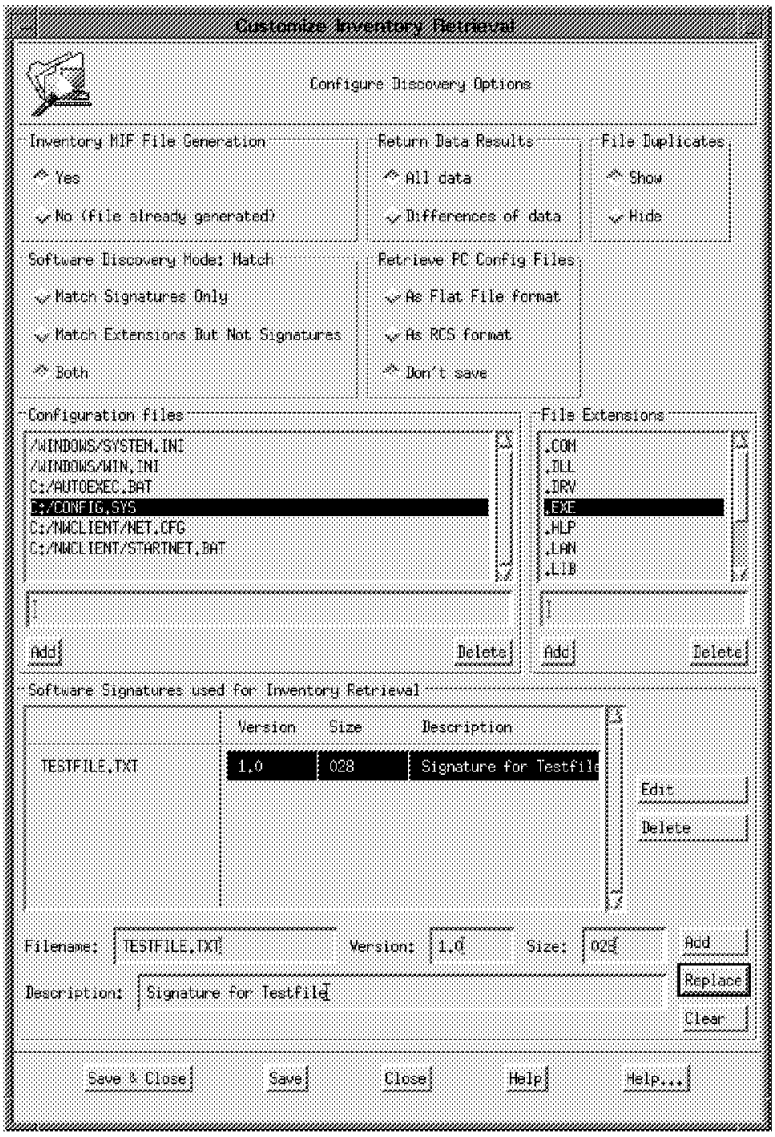


Figure 201. Customize Inventory Retrieval (Add Software Signature)

Customizing the profile by adding names as software signatures to be used for retrieval will change the useradd.ini file at the server side (see Figure 202).

```
MIFPATH=\TIVOLI\SCAN\OUTPUT
ScanExtensions=.EXE .COM .SYS .NLM .DLL .DRV .LIB .HLP .LAN .ORI
&lbracket.Applications&rbracket.
<I>,TESTFILE.TXT,028,Signature for Testfile,1.0
Mode=All
Duplicate=ON
CfgFiles1=C:\CONFIG.SYS C:\NWCLIENT\STARTNET.BAT
CfgFiles2=C:\AUTOEXEC.BAT C:\NWCLIENT\NET.CFG
CfgFiles3=\WINDOWS\WIN.INI
CfgFiles4=\WINDOWS\SYSTEM.INI
```

Figure 202. The useradd.ini Script

6.3.3.3 The Scanning Process on the UNIX Managed Nodes

Inventory information of a managed node is scanned by the UNIX inventory scanning component. The inventory scanning program on UNIX managed nodes is not the same as the program on PC managed nodes. Unlike the PC managed node scanning process, the UNIX scanning process is very limited.

The UNIX scanning program is sent to the UNIX managed node by the distribute inventory profile function. The UNIX scanning process does not create a MIF file, does not scan for software, and only returns information about the hardware configuration of the UNIX node. The scanning process of UNIX managed nodes is done by the SysInfo package. The information returned by SysInfo is dependent on the architecture of the managed node. The hardware inventory information is sent back to the inventory server and stored in the TME configuration repository.

6.3.4 Viewing the Scanned Data

We can view the TME configuration repository records that represent the inventory information of each managed system. The information that is available with the GUI is only part of the inventory information kept in the database. To see all the information that Inventory stores, you can write your own SQL scripts and run them against the RDBMS, or you can use the Query Facility. See 6.5, “TME Query Facility” on page 235 for details.

The administrator has to have the Inventory_view and senior or super roles set to be able to view the software and hardware inventory information of any subscriber.

6.3.4.1 Viewing the Scanned Data of PC Managed Nodes

To view the hardware inventory of PC managed nodes you have to click on that node with the right mouse button and scroll down to the Hardware Inventory... menu. As shown in Figure 203 on page 232, information on architecture, physical memory, paging space, processor model, processor speed, operating system, and version of operating system is displayed. There is no display of built-in adapters and devices, disks or free disk space.

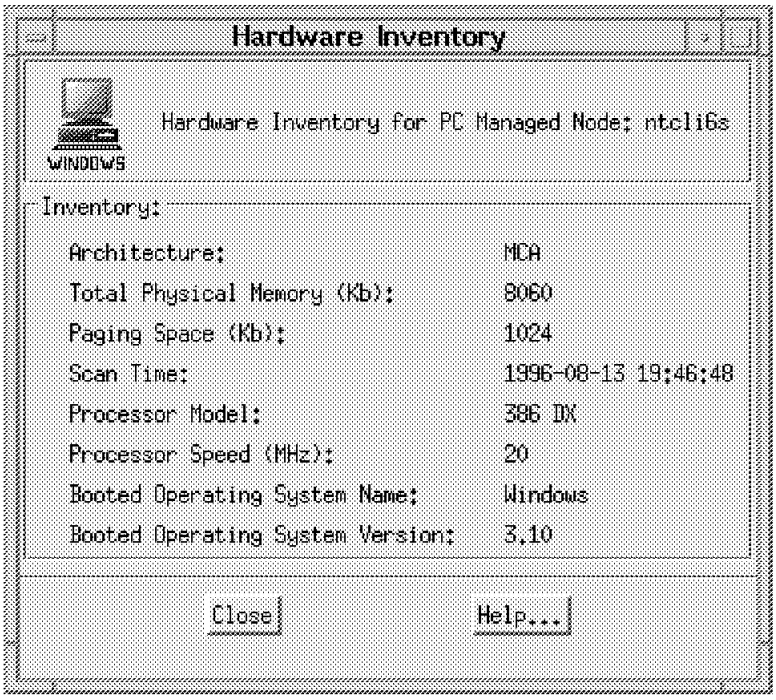


Figure 203. Hardware Inventory (Windows 3.10)

To view the software inventory of PC managed nodes, you can click on that node with the right mouse button and scroll down to the Software Inventory.... menu. Figure 204 shows how to select Software Inventory from the pop-up menu.

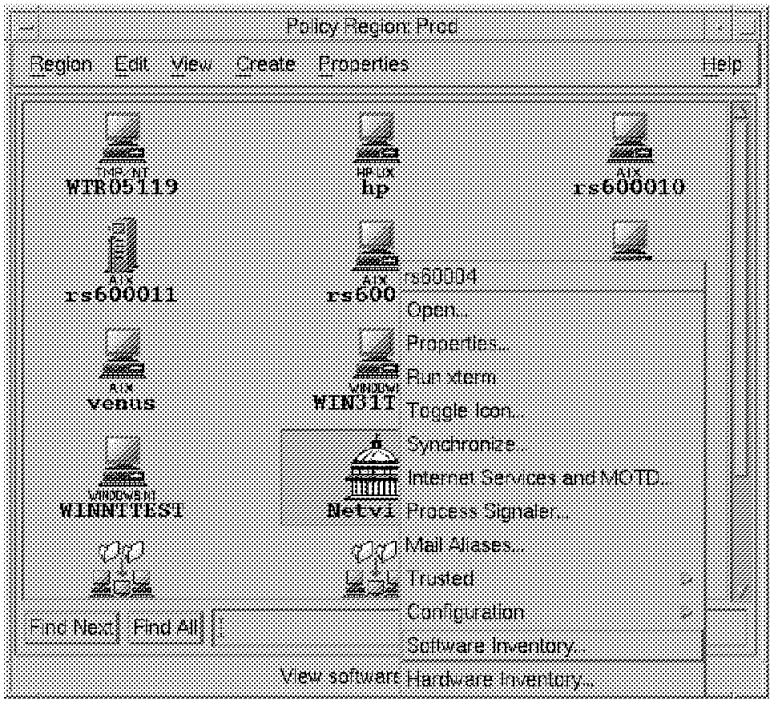


Figure 204. Select Software Inventory

Release the mouse button to view the information shown in Figure 205 on page 233.

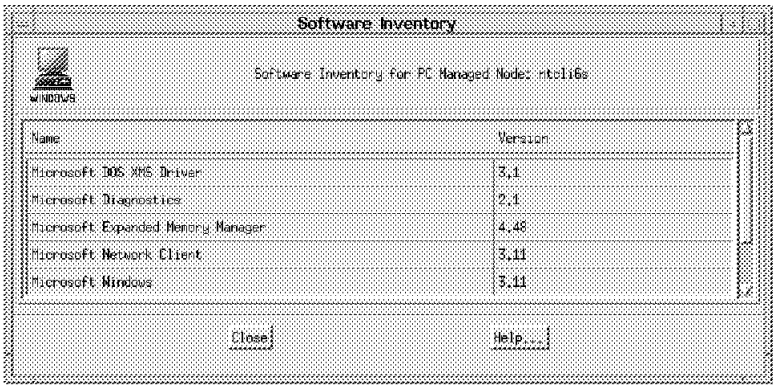


Figure 205. Software Inventory (Windows 3.10)

6.3.4.2 Viewing the Scanned Data of UNIX Managed Nodes

To view the hardware inventory of UNIX managed nodes, you have to click on that node with the right mouse button and scroll down to the Hardware Inventory... menu. Release the mouse button to view the information shown in Figure 206. Information on architecture, physical memory, paging space, operating system, and version of operating system is displayed. The information about physical volumes, logical volumes and filesystems is stored in the TME repository database. You can display this information with a user-written query to the database.

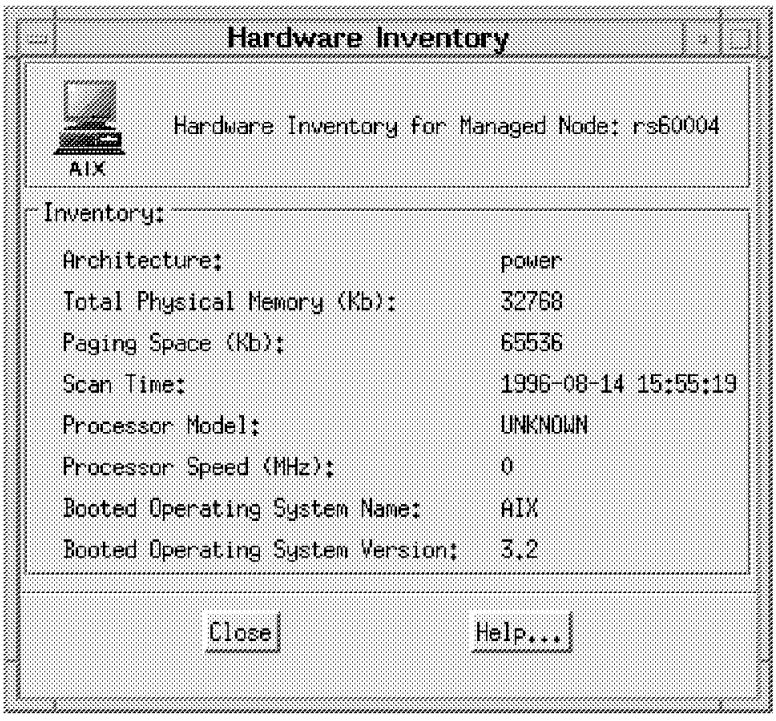


Figure 206. Hardware Inventory (AIX)

No information on installed software products and applications for AIX, HP-UX, and SunOS operating systems is displayed at this time.

6.4 Information without Using Tivoli/Inventory

For a workstation to be a managed node or PC managed node, you have to define the node to a policy region. See 2.4.3, “How to Start the Tivoli Management Environment” on page 28 and 2.4.5, “Installing PC Managed Node as TME Client” on page 31. When you define a managed node or PC managed node, the agent that has been defined to TME determines the properties of the workstation and returns the information to the TMR server.

You can display this information by selecting a subscriber in a profile manager using the right mouse button. Scroll down to Properties... and release the mouse button. Figure 207 shows the display for PC managed nodes, and Figure 208 on page 235 for managed nodes.

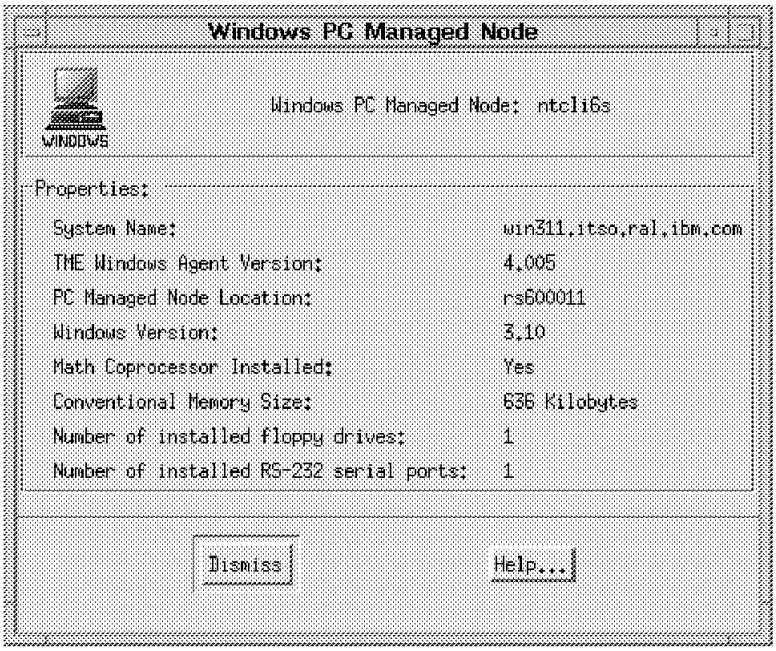


Figure 207. PC Managed Node Properties

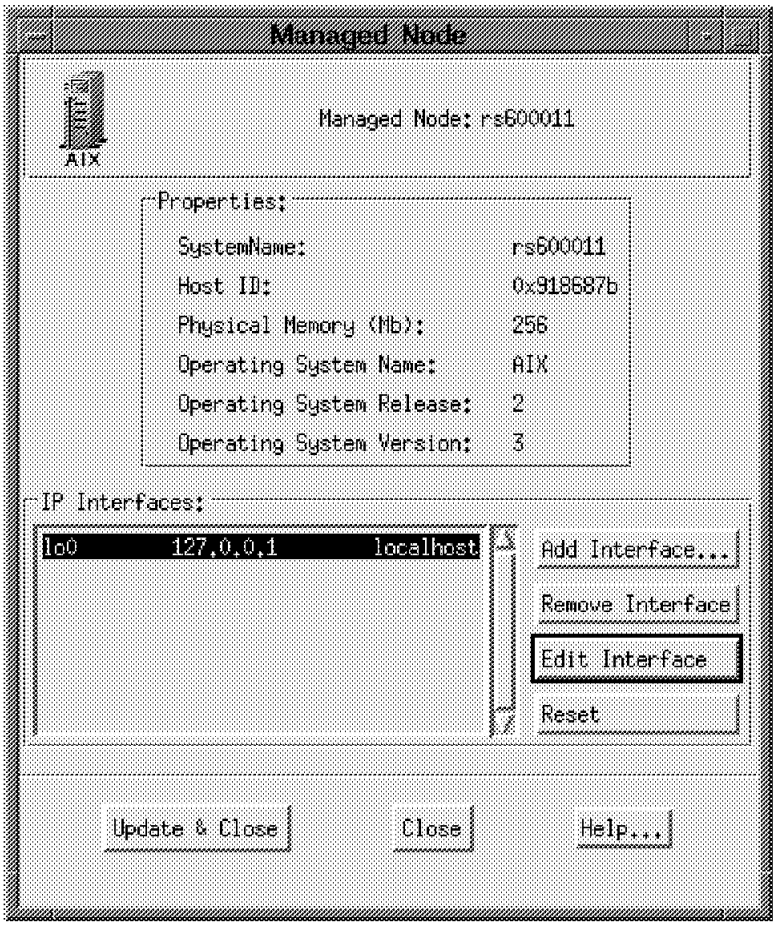


Figure 208. Managed Node Properties

6.5 TME Query Facility

The TME query facility is part of Tivoli/Inventory and provides you with easy access to the inventory database to dynamically generate a set of subscribers to a profile.

The TME query facility enables you to create query libraries, which contain individual queries that you define depending on your needs. These queries run against the TME configuration repository. They are most useful if you run out of a profile manager to select, for example, the distribution targets of a Tivoli/Courier file package distribution.

All Systems Must Be Scanned

TME queries retrieve the information from the inventory database. Therefore, they will only select subscribers that have previously been scanned.

If you want to select subscribers by dynamically changing criterions such as free disk space, initiate a scanning process to update the inventory database before you run the query.

6.5.1 Create Query Libraries

TME queries reside in TME query libraries. Query libraries can be used to organize the queries within the TME. For example, you can create query libraries that contain all queries for a specific platform or for a geographic area.

The following example creates a query library called Qlib_AIX in policy region Raleigh:

1. Login as root or as any other TME administrator with the senior role.
2. Bring up the Tivoli desktop by entering `tivoli`. Refer to 2.4.3, "How to Start the Tivoli Management Environment" on page 28 for more information.
3. Double-click on the policy region icon where you plan to create the query library.
4. Select **Create = > QueryLibrary** and fill in the name of the new query library in the Create Query Library dialog as shown in Figure 209.

 If there is no Query Library entry on the Create menu, you have to add Query Library as a managed resource type for the policy region. This is done by double-clicking on the Policy Region icon, then select **Properties => Managed Resources** and move the Query Library entry from the Available Resources to the Current Resources area. Refer to Figure 10 on page 29 for an example of the Managed Resources dialog.
5. Select **Create & Close**.



Figure 209. Create Query Library

Command Line to Create Query Library: The command line to create a query library called Qlib_AIX in the policy region Raleigh is:

```
wcrtqlib Raleigh Qlib_AIX
:i1.wcrtqlib
```

6.5.2 Create Queries

Since TME queries reside in TME query libraries, you have to create the query library before you are able to make a new query. Refer to 6.5.1, “Create Query Libraries” on page 236 for information on how to create query libraries.

Queries are run against the configuration repository and are used to select a set of subscribers to a profile manager.

The following example creates the query All_AIX_32 in query library Qlib_AIX. Query All_AIX_32 selects all systems that run AIX Version 3.2. The equivalent SQL statement is:

```
select TME_OBJECT_LABEL from INVENTORYDATA
where BOOTED_OS_NAME='AIX' and
      BOOTED_OS_VERSION='3.2';
```

1. Log in as root or as any other TME administrator with the senior role.
2. Bring up the Tivoli desktop by entering tivoli. Refer to 2.4.3, “How to Start the Tivoli Management Environment” on page 28 for more information.
3. Double-click on the icon that represents the query library you selected to contain the new query.
4. Select **Create = > Query** and fill in the Create Query dialog as described below and as shown in Figure 210.

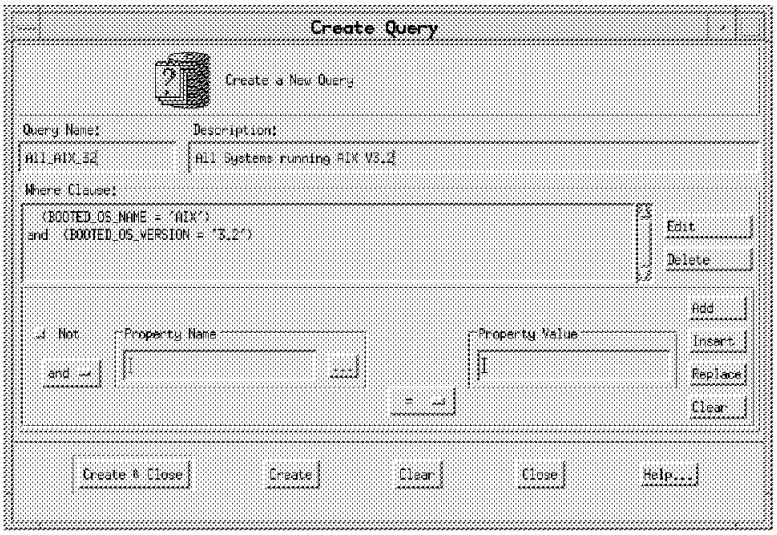


Figure 210. Create Query

- Enter a name for the query in the Query Name field.
- The Description field is optional, and is used to briefly describe the query.
- Start building the query by entering a database column name into the Property Name field. You can choose any field by selecting the **...** (ellipses) button.

 For our example, we don’t select an operator from the and field (because it is the first part of the query). We select **BOOTED_OS_NAME** from the Inventory Properties window as shown in Figure 211 on page 238.

- Choose a logical operator from the **=** field to establish a relationship between the name and the value of the property.

 Remember if you select the **LIKE** operator, the corresponding SQL wildcard is the % character.

 For our example, we choose the **=** (equal) operator.

- The Property Value field contains the query value.

 For our example, we entered AIX.

- Click on the **Add** button on the Create Query dialog to add these options to the Where Clause field.

- Repeat the previous steps for each search clause. When adding a new clause, you can use the logical **and**, **or** and **Not** operators to build a compound query.

 For our example, we selected **and**, the database column name **BOOTED_OS_VERSION** for the Property Name field, the equal operator (**=**) to relate the property name to its value, and finally entered 3.2 into the Property Value field.

5. Select **Create & Close**.

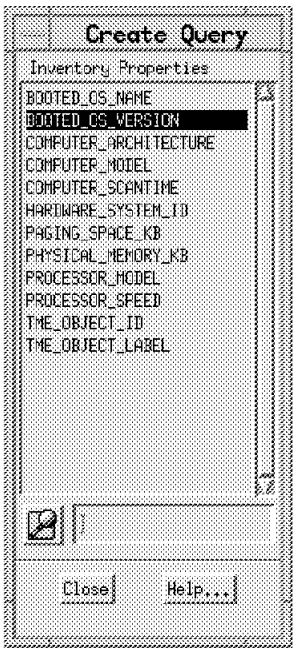


Figure 211. Inventory Properties

6.5.3 Extend the Query Facility

The query facility checks and runs its queries by default against the INVENTORYDATA view. This view consists of default fields shown in Figure 212 on page 239.

```
SQL> desc INVENTORYDATA
 Name                           Null?    Type
 ------------------------------ -------- ----
 HARDWARE_SYSTEM_ID             NOT NULL VARCHAR2(64)
 TME_OBJECT_ID                           VARCHAR2(128)
 TME_OBJECT_LABEL                        VARCHAR2(32)
 COMPUTER_ARCHITECTURE                   VARCHAR2(32)
 COMPUTER_MODEL                          VARCHAR2(64)
 PHYSICAL_MEMORY_KB                      NUMBER(38)
 PAGING_SPACE_KB                         NUMBER(38)
 COMPUTER_SCANTIME                       VARCHAR2(32)
 PROCESSOR_MODEL                         VARCHAR2(32)
 PROCESSOR_SPEED                         NUMBER(38)
 BOOTED_OS_NAME                          VARCHAR2(16)
 BOOTED_OS_VERSION                       VARCHAR2(16)
```

Figure 212. Fields in INVENTORYDATA View

If you want to select subscribers by additional fields that are not part of the default INVENTORYDATA view, you have the following choices:

- Extend the existing INVENTORYDATA view
- Define queries that don't use the INVENTORYDATA view

6.5.3.1 Extend the INVENTORYDATA View

If you choose to create your queries using the query facility's GUI, you have to extend the INVENTORYDATA view to suit your requirements. These new database fields you create in the INVENTORYDATA view are available immediately to the query facility, and the available selections in the Inventory Properties window of the Create Query dialog, as shown in Figure 211 on page 238, are updated dynamically with the new database fields.

Beware - INVENTORYDATA View May Change

Along with the further development of Tivoli/Inventory, the data model and the INVENTORYDATA view may change. So be aware that all of your changes might become obsolete with later releases of Tivoli/Inventory.

The code in Figure 213 on page 240 shows how we extended the original INVENTORYDATA view by three fields from the table LOGICALDRIVE.

```
rem *****************************************************************
rem Description: Deletes : INVENTORYDATA
rem              Creates : INVENTORYDATA
rem              Adds    : LOGICALDRIVE_NAME
rem                        LOGICALDRIVE_MOUNTDIR
rem                        LOGICALDRIVE_SIZE_KB
rem *****************************************************************

drop view INVENTORYDATA;

create view INVENTORYDATA as
(
select
      COMPUTER_VIEW.HARDWARE_SYSTEM_ID,
      TME_OBJECT_ID,
      TME_OBJECT_LABEL,
      COMPUTER_ARCHITECTURE,
      COMPUTER_MODEL,
      PHYSICAL_MEMORY_KB,
      PAGING_SPACE_KB,
      COMPUTER_SCANTIME,
      PROCESSOR_MODEL,
      PROCESSOR_SPEED,
      BOOTED_OS_NAME,
      BOOTED_OS_VERSION,
      LOGICALDRIVE.LOGICALDRIVE_NAME,
      LOGICALDRIVE.LOGICALDRIVE_MOUNTDIR,
      LOGICALDRIVE.LOGICALDRIVE_SIZE_KB
from  COMPUTER_VIEW,
      MEMORY_VIEW,
      PROCESSOR_VIEW,
      LOGICALDRIVE
where MEMORY_VIEW.HARDWARE_SYSTEM_ID=COMPUTER_VIEW.HARDWARE_SYSTEM_ID and
      PROCESSOR_VIEW.HARDWARE_SYSTEM_ID=COMPUTER_VIEW.HARDWARE_SYSTEM_ID and
      LOGICALDRIVE.HARDWARE_SYSTEM_ID=COMPUTER_VIEW.HARDWARE_SYSTEM_ID
);

commit;
quit
```

Figure 213. Extended INVENTORYDATA View

6.5.3.2 Run Queries against Other Views

Queries that don't run against the INVENTORYDATA view cannot be created using the TME desktop. They have to be created using the command line interface. The definition task consists of three steps:

1. Create the new database view.

 Refer to 6.5.3.1, "Extend the INVENTORYDATA View" on page 239 for an example.

2. Create the query.

 The following example creates a query called all_Windows in query library Qlib_PC. The description for this query is PCs running Windows 3.11, and it selects all PC managed nodes running Microsoft Windows, Version 3.11.

```
wcrtquery -d 'PCs running Windows 3.11' \
:i1.wcrtquery
-w "(SOFTWARE_ID='Microsoft Windows') and \
(SOFTWARE_VERSION_ID='3.11')"  Qlib_PC all_Windows
```

3. Set the desired view for the query.

 This example assigns the newly created BIG_VIEW to query all_Windows:

   ```
   wsetquery -v BIG_VIEW all_Windows
   :i1.wsetquery
   ```

You can now run the query to select a set of subscribers to a profile manager, just as any other query that was created using the graphical user interface.

6.6 TME Configuration Repository

Before Tivoli/Inventory can scan and store configuration data, the TME configuration repository must be set up. This means that there must be a database instance configured with the *TME configuration repository scheme*. The TME configuration repository scheme is a script containing SQL statements, that creates a set of tables, views, etc. designed to work in conjunction with Tivoli/Inventory. For information on installing and setting up the configuration database, refer to 6.2.1, “Installing the RDBMS” on page 207.

This section provides you with more information on the database model, how to access this data and how to extend the repository data model.

6.6.1 TME Configuration Repository Data Model

The TME configuration repository provides you with a structure that permits you to store the following data that is related to the configuration of your computer infrastructure:

- Computer hardware information, such as processor type, keyboard type, physical memory size, types of storage devices, etc.
- System software information, such as system BIOS, operating system, network addresses (IP address) and configuration files
- Systems and applications management information, such as TME tasks and file package definitions
- Physical information, such as the name of the person responsible for a system, the location of the system, etc.

Most of the time, an administrator accesses the data in the configuration repository by means of the query facility. The queries created via the query facility run by default against the INVENTORYDATA view. The INVENTORYDATA view contains those fields of the configuration database that are most commonly used to select a set of subscribers to a profile manager. Although the scanning processes for the different platforms don't yet return all possible configuration information, there is much more relevant configuration information available in the database. To fully exploit all of the configuration information kept in the configuration repository, you need to have a good understanding of the configuration repository data model. Refer to Appendix A, “TME Configuration Repository” on page 519, or to the *Tivoli/Inventory User's Guide* for more information on the tables and views that make up the TME configuration repository scheme.

6.6.2 Enterprise Specific Configuration Data

On a PC managed node the inventory scanning program creates a MIF file as described in 6.3.3.2, “The Scanning Process on PC Managed Nodes” on page 228. You can add your own requirements in the form of groups to the MIF file or create a new MIF file by hand. This file will be sent to the server and the data will be put into the database.

If you want to create your own groups, be careful how you name them and how you create the tables at the configuration data model. Be aware of the following rules:

- Do not modify the definition of existing tables in the Tivoli scheme.
- For every group of information you add to the MIF file create your own table in the database.
- To relate the table to an existing table in Tivoli, make the primary key of the existing table the foreign key in your table.
- Every new group in a customized MIF file must correspond to a table in the database scheme.
- The MIF group name must be identical to the name of the database table.
- The MIF attributes in a group must be identical to the column names in the table.
- The column types must match the attribute types.
- An attribute must occur only one time in a MIF file.
- To track the configuration change history of a custom table, it should be related to the CONFIG_CHANGE_HISTORY table as evident in existing Tivoli tables that have been tracked.

At this time, there is no possibility to extend the inventory of UNIX managed nodes, as there is no MIF file created to get the inventory to the database.

6.7 Create a Reference Model

Once the initial system configuration information is stored in the TME configuration repository, you can create a set of database records that represent the ideal system configuration as a reference model for all other systems.

To create a reference model, you can either find a system with a configuration that models the ideal configuration, or you can define your reference model yourself.

In our example, we define a simple reference model from scratch, then add it to the TME configuration repository.

6.7.1 Add a Software Reference Model

The shell script add_ref_ora.sh, as shown in Figure 214 on page 243, creates a reference model called Ref_Excel. This reference model indicates the presence of the Microsoft Excel, Version 7.0 application.

Note: There is an example on how to create a reference model in the *Tivoli/Inventory User's Guide*. This example is only applicable if your configuration repository is based on the Sybase RDBMS, and does not

reflect all the constraints that are applied to the database scheme in our Oracle-based environment.

```
#!/bin/ksh
# ********************************************************************
# @(#) Add Reference Model to Configuration Repository
# ********************************************************************

# Create Reference Model for Excel
sqlplus -s tivoli/tivoli <<EOF
insert into HARDWARE_SYSTEM
(HARDWARE_SYSTEM_ID)
values ('Ref_Excel')
;
insert into COMPUTER_SYSTEM
(HARDWARE_SYSTEM_ID, IS_A_REFERENCE_SYSTEM)
values ('Ref_Excel', 1)
;
insert into SOFTWARE
(SOFTWARE_ID)
values ('Microsoft Excel')
;
insert into SOFTWARE_VERSION
(SOFTWARE_VERSION_ID, SOFTWARE_ID, VERSION_LANGUAGE_EDITION)
values ('7.0', 'Microsoft Excel', 'English')
;
insert into INSTALLED_SOFTWARE_VERSION
(SOFTWARE_VERSION_ID, SOFTWARE_ID, HARDWARE_SYSTEM_ID,
VERSION_LANGUAGE_EDITION, CONFIG_CHANGE_TYPE,
INSTALLED_VERSION_PATH)
values ('7.0', 'Microsoft Excel', 'Ref_Excel',
'English', 'insert', 'c:\MS')
;
commit;
quit
EOF
```

Figure 214. The add_ref_ora.sh Script

To define a reference model, complete the following steps:

1. Create an entry for it in the HARDWARE_SYSTEM table. The name for the reference model is stored as a HARDWARE_SYSTEM_ID.
2. In the COMPUTER_SYSTEM table, define this HARDWARE_SYSTEM_ID (reference model name) as a reference system by updating the IS_A_REFERENCE_SYSTEM field.
3. Register the name of the application as a SOFTWARE_ID in the SOFTWARE table.
4. Enter the values for the SOFTWARE_VERSION_ID, SOFTWARE_ID and VERSION_LANGUAGE_EDITION into the respective fields of the SOFTWARE_VERSION table.
5. Finally, enter the values for SOFTWARE_VERSION_ID, SOFTWARE_ID, HARDWARE_SYSTEM_ID, VERSION_LANGUAGE_EDITION, CONFIG_CHANGE_TYPE, and the INSTALLED_VERSION_PATH into the INSTALLED_SOFTWARE_VERSION table.

6.7.2 Assign a Reference Model to a System

The shell script add_ref_system.sh, as shown in Figure 215, assigns a specific system to the reference model Ref_Win311.

```
#!/bin/ksh
# ************************************************************************
# @(#) Adds a System to the REFERRED_SYSTEMS Table
# ************************************************************************

if [[ $# -ne 1 ]]; then
   echo "Usage: $(basename $0) TME_OBJECT_LABEL"
   exit 1
fi

SYSTEM_ID=$1

# Get HARDWARE_SYSTEM_ID
VAR=`sqlplus -s tivoli/tivoli <<EOF | grep -E -v "HARDWARE|---|selected"
select HARDWARE_SYSTEM_ID
from COMPUTER_SYSTEM
where TME_OBJECT_LABEL='$SYSTEM_ID'
;
quit
EOF`

# Some String Fiddling
VAR=`echo $VAR | sed -e 's/\\n//g'`

# Update REFERRED_SYSTEMS
sqlplus -s tivoli/tivoli <<EOF
insert into REFERRED_SYSTEMS
(HARDWARE_SYSTEM_ID, REFERRED_SYSTEM_ID)
values ('$VAR', 'Ref_Win311')
;
commit;
quit
EOF
```

Figure 215. The add_ref_system.sh Script

6.7.3 Compound Reference Models

The examples in this sections show you how to create small reference models from scratch and how to assign them to individual systems. But there are also other ways to define reference models:

- It is possible to define one HARDWARE_SYSTEM_ID as a reference model and assign various products to it.

 For example, you can define a reference model containing all components representing the ideal configuration for systems belonging to the Admin department.

- If you have a previously scanned system that matches your ideal software configuration, you can update this system's entry in the COMPUTER_SYSTEM table, defining it as a reference model in the IS_A_REFERENCE_SYSTEM field.

You can then use this HARDWARE_SYSTEM_ID and assign it to various systems in the REFERRED_SYSTEMS table.

6.7.4 Change Notification

Once a reference model exists in the configuration repository, you can compare it with the records in the repository that represent the configuration of all the systems in the enterprise.

This section describes how to create and use scripts that list the actual system records and the records of the reference models, compare the output of the two and provide you with a list of the differences found.

This example can be expanded to take almost any action that you define when differences are found. For our example, we created three scripts:

- get_diff_sw.sh
- list_inst_software.sh
- list_ref_software.sh

The get_diff_sw.sh script, as shown in Figure 216 on page 246, acts as the main script for our change notification. It requires the TME_OBJECT_LABEL to be supplied as a parameter and derives the required HARDWARE_SYSTEM_ID from the corresponding TME_OBJECT_LABEL. (Look at the format of the HARDWARE_SYSTEM_ID and you'll agree that this feature comes in quite handy.)

It then calls list_inst_software.sh and list_ref_software.sh to get a list of the installed software and the reference model software for the system specified by the supplied TME_OBJECT_LABEL. It then compares the output of these two lists and reports the differences.

```
#!/bin/ksh
# ********************************************************************
# @(#) Reports Difference for a specific Client
# ********************************************************************

if [[ $# -ne 1 ]]; then
   echo "Usage: $(basename $0) TME_OBJECT_LABEL"
   exit 1
fi

SYSTEM_ID=$1

TMP_DIR=/tmp/$(basename $0)
LIST_INST=./list_inst_software.sh
LIST_REF=./list_ref_software.sh

LIST_INST_FILE=$TMP_DIR/list_inst_file
LIST_REF_FILE=$TMP_DIR/list_ref_file
DIFF_FILE=$TMP_DIR/diff_file

mkdir -p $TMP_DIR

# Get HARDWARE_SYSTEM_ID
VAR=`sqlplus -s tivoli/tivoli <<EOF | grep -E -v "HARDWARE|---|selected"
select HARDWARE_SYSTEM_ID
from COMPUTER_SYSTEM
where TME_OBJECT_LABEL='$SYSTEM_ID'
;
quit
EOF`
VAR=`echo $VAR | sed -e 's/\\n//g'`

$LIST_INST "$VAR" |sed -e 's/\\n//g' >$LIST_INST_FILE
$LIST_REF "$VAR" |sed -e 's/\\n//g'  >$LIST_REF_FILE

diff $LIST_INST_FILE $LIST_REF_FILE >$DIFF_FILE

#Format Output
echo "=============================================================="
echo "Differences for: $SYSTEM_ID "
echo "=============================================================="
echo "Additional:"
echo "----------"
echo "$(cat $DIFF_FILE | sed -e '/< $/d' | grep "<")"
echo "\n"
echo "Missing:"
echo "-------"
echo "$(cat $DIFF_FILE | sed -e '/< $/d' | grep ">")"
echo "=============================================================="

rm -rf $TMP_DIR
```

Figure 216. The get_diff_sw.sh Script

The list_inst_software.sh script, as shown in Figure 217 on page 247, lists all installed software for one system. It is normally called up by the script get_diff_sw.sh, but can also be used autonomously, provided you supply the HARDWARE_SYSTEM_ID when calling it up.

```
#!/bin/ksh
# ********************************************************************
# @(#) Lists all Installed Software for a specific System
# ********************************************************************

if [[ $# -ne 1 ]]; then
   echo "Usage: $(basename $0) HARDWARE_SYSTEM_ID"
   exit 1
fi

SYSTEM_ID=$1

sqlplus -s tivoli/tivoli <<EOF | grep -E -v "SOFTWARE|---|rows selected"
select SOFTWARE_ID, SOFTWARE_VERSION_ID
from   INSTALLED_SOFTWARE_VERSION
where  (INSTALLED_SOFTWARE_VERSION.HARDWARE_SYSTEM_ID='$SYSTEM_ID')
;
quit
EOF
```

Figure 217. The list_inst_software.sh Script

The list_ref_software.sh script, as shown Figure 218, lists all software contained by the reference model(s) assigned to this system. It is normally called up by the script get_diff_sw.sh, but can also be used autonomously, provided you supply the HARDWARE_SYSTEM_ID when calling it up.

```
#!/bin/ksh
# ********************************************************************
# @(#) Lists all Reference Software for a specific System
# ********************************************************************

if [[ $# -ne 1 ]]; then
   echo "Usage: $(basename $0) HARDWARE_SYSTEM_ID"
   exit 1
fi

SYSTEM_ID=$1

sqlplus -s tivoli/tivoli <<EOF | grep -E -v "SOFTWARE|---|rows selected"
select SOFTWARE_ID, SOFTWARE_VERSION_ID
from   INSTALLED_SOFTWARE_VERSION
where  (INSTALLED_SOFTWARE_VERSION.HARDWARE_SYSTEM_ID in
       (select REFERRED_SYSTEM_ID from REFERRED_SYSTEMS
        where (REFERRED_SYSTEMS.HARDWARE_SYSTEM_ID='$SYSTEM_ID')))
;
quit
EOF
```

Figure 218. The list_ref_software.sh Script

Chapter 7. Tivoli/Courier and Tivoli/Inventory Interoperability

To complete the configuration of the software distribution environment, you should consider the installation of other Tivoli applications, such as Tivoli/Inventory, to provide additional capabilities to Tivoli/Courier. Using Tivoli/Inventory, you can query a configuration database to determine targets of a distribution.

For example, you can use queries to select machines with a certain operating system and version, select machines with a particular application installed on them, or select machines with specific hardware configurations.

In 5.4.4, "Software Distribution of MQSeries for AIX" on page 159, we show how to distribute the MQSeries for AIX product. Since this package is valid only for AIX 4.1, we could build a query that would select only machines with AIX 4.1 installed, as targets for the distribution.

7.1 Create Tivoli/Inventory Query

First, we created a Query library that will contain queries to determine the level of operating system installed on a machine. Because we want to use these queries for software distribution, we created the query library in the SoftDist policy region.

We created a query library and called it Software Queries. Refer to 6.5.1, "Create Query Libraries" on page 236 for details on how to create the query library.

Once the query library is created, you can create the query All_AIX_41 to select all machines that run AIX Version 4.1. Refer to 6.5.2, "Create Queries" on page 237 for details on how to create a query.

In our project, we created a query where BOOTED_OS_NAME='AIX' and BOOTED_OS_VERSION='4.1'.

This query will scan the configuration repository for machines that satisfy those criteria. The repository is a snapshot of the network, created by Tivoli/Inventory before the query is run.

7.2 Use Query in Software Distributions

After the query is created, you can perform software distribution and use the query facility to select only the subscribers that have AIX 4.1 installed.

For detailed steps on distributing the MQSeries, refer to 5.4.4.1, "Distribute the File Package for MQSeries" on page 160.

1. Select **File Package = > Distribute** from the **AIX_mqm^2.1** File Package icon to display the Distribute File Package window (see Figure 219 on page 250).

© Copyright IBM Corp. 1996

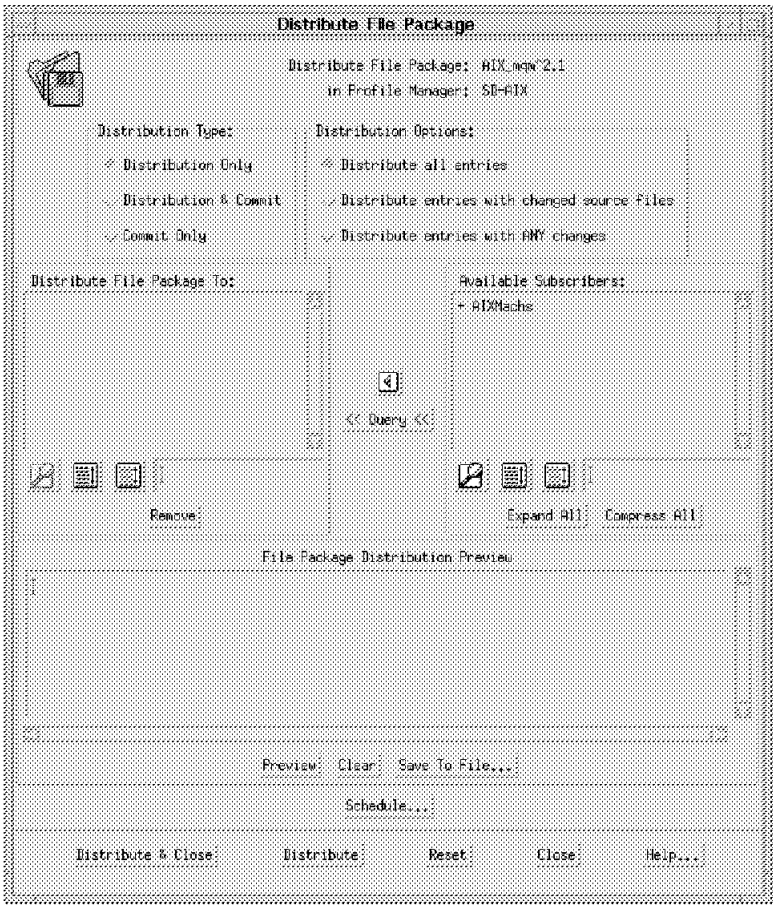


Figure 219. Distribute File Package (MQSeries)

2. Select the **Query** button to display the Execute a Query dialog (see Figure 220).

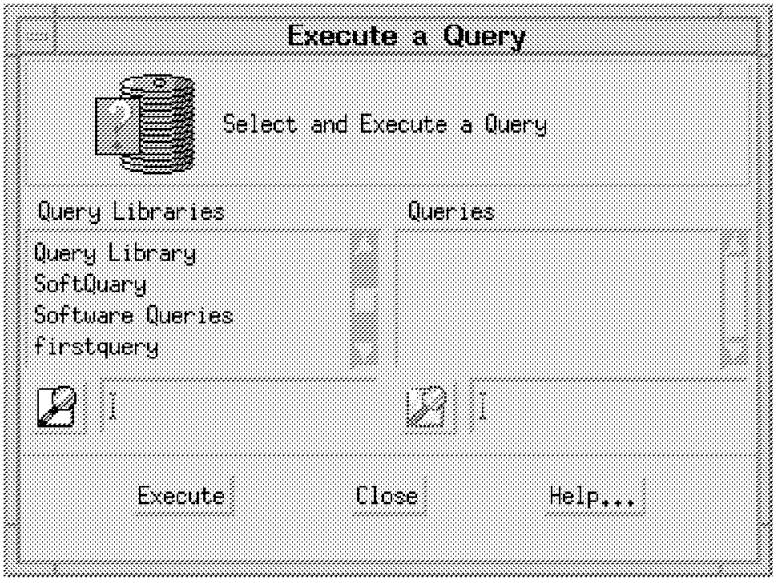


Figure 220. Execute a Query

This dialog lists query libraries previously created using Tivoli/Inventory.

3. Select **Software Queries** to display the queries.

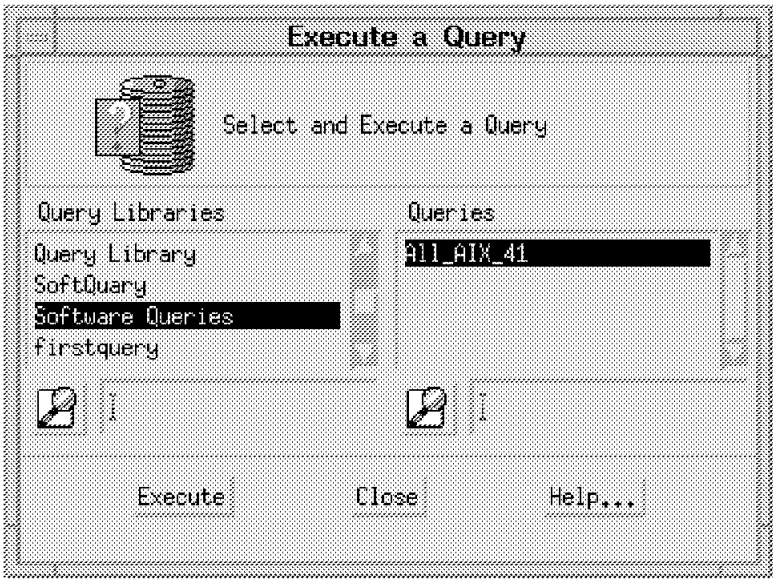


Figure 221. Execute a Query

4. To execute the query, select **All_AIX_41** and **Execute**.

 Tivoli/Inventory queries the repository and returns the machines that satisfy the query (in our case, mercury). Then it automatically adds these machines to the Distribute File Package To list box.

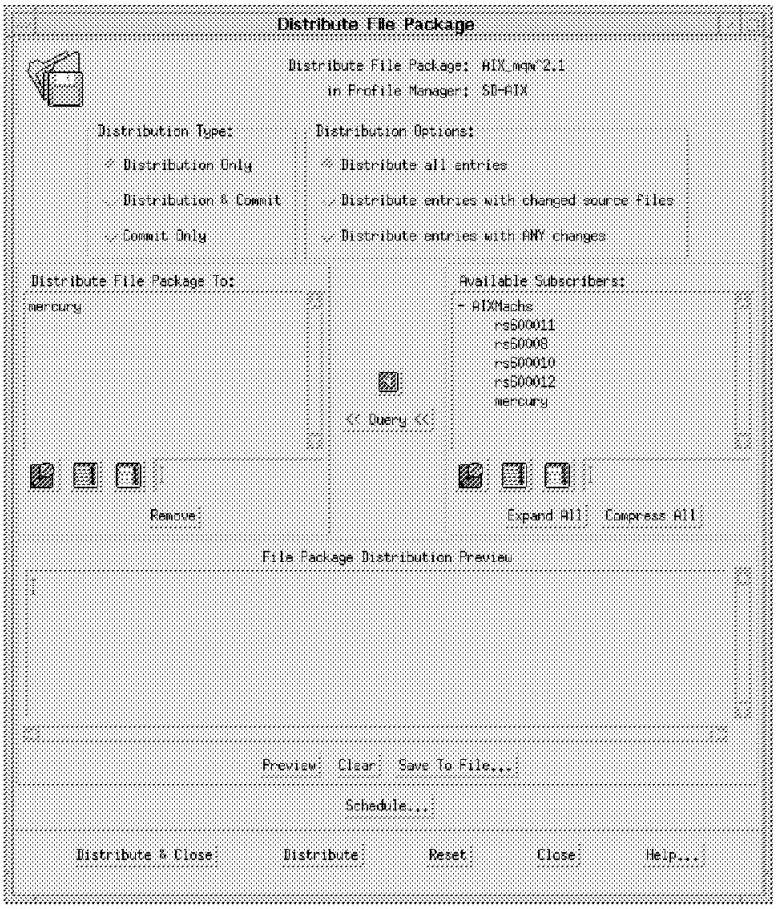


Figure 222. Distribute File Package (MQSeries with Subscribers)

5. You can now select the **Distribute & Close** button to begin distributing the file package to the target mercury, which is an AIX 4.1 system.

Chapter 8. TME 10 User Administration

TME 10 User Administration was previously known as Tivoli/Admin, we will use the former name for the product in this chapter. It is the third deployment application among the Tivoli core applications. You may of Tivoli/Courier as a *generic* application for data distribution and remote command execution. By constrast, Tivoli/Admin is a *specialized* application for the configuration of particular aspects of a remote system.

Tivoli/Admin provides central management of the following resources:

- Users and groups
- Host configurations (including NIS)
- Mail aliases
- A number of standard system configuration files

In this chapter we concentrate mainly on user management.

Before working with Tivoli/Admin you should first read Chapter 1, ″Introduction to Tivoli User and Group″ in the *Tivoli User and Group Management Guide*. This will introduce you to the concepts of Tivoli/Admin, including:

- How Tivoli users and groups work
- User and group account components and profiles
- User and group profile policy
- Populate, subscribe, distribute
- Password control
- The User Locator
- Individual record locking

In this chapter we illustrate some of these concepts with examples of how to create, modify and distribute user profiles and how to manage user records. Note, however, that we do not reproduce all of the information contained in the *Tivoli User and Group Management Guide*.

8.1 Planning, Installation and Initial Configuration

As you would expect, Tivoli/Admin makes use of the base Tivoli functions such as management by subscription. This means that you should give careful thought to planning the installation before you implement it in a production environment. In practical terms this means you should do two things:

1. Create a policy region structure that maps to the adminstrative organization of your managed environment.
2. Create a Profile Manager structure that maps to the application organization of the managed nodes.

Of course, this planning process is not application-specific. You are likely to want to use a number of Tivoli applications within the one policy region/profile manager design.

© Copyright IBM Corp. 1996

8.1.1 Installing Tivoli/Admin

The Tivoli/Admin installation process is exactly the same as for installing any other Tivoli product on top of the base platform. We described this in 2.5, “Product Installation” on page 40.

The resources that Tivoli/Admin manages are represented in the Tivoli environment by a number of profile types. When you install Tivoli/Admin you will have the following new profile types available:

User Profiles These contain information related to users, such as user names, user IDs, passwords, home directories and login shells.

Group Profiles These mirror information in the UNIX /etc/groups file, such as valid groups, group IDs and membership lists.

Host Namespace Profiles These contain information about IP name/address mapping, mirroring the entries found in the /etc/hosts file.

Before you can use the profiles you need to add these resources to the current resources of the Policy Regions in which you want to place Admin profiles.

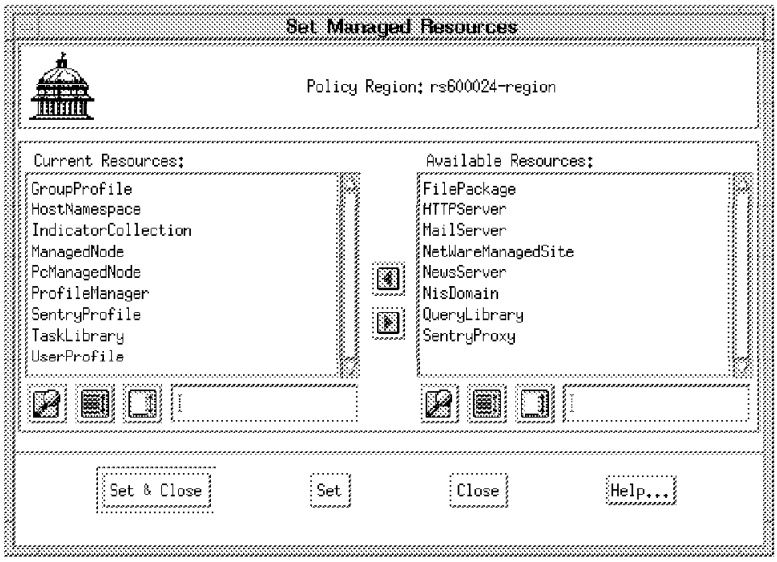


Figure 223. Add the Admin Resources to the Policy Region

To add the resources from the command line, enter the following sequence of instructions:

```
wsetpr UserProfile @PolicyRegion:rs600024-region
wsetpr GroupProfile @PolicyRegion:rs600024-region
wsetpr HostNamespace @PolicyRegion:rs600024-region
```

8.2 User ID Management Example

To illustrate the use of Tivoli/Admin for defining and administering user IDs, we use a practical example.

8.2.1 Setting the Scene

The environment we use in this example is set up as follows:

- The policy region is rs600024-region.
- The managed nodes are all AIX systems:

 rs600019
 rs600020
 rs600021
 rs600024

- There are two sets of users on these machines. We set up different defaults for each type, so they are put in separate user profiles.

 Residents Short-term users of the systems

 Assignees Long-term users of the systems

- Both types of users exist on all managed nodes.

8.2.2 Creating User Profiles

The first step is to create a Profile Manager. We create only one Profile Manager with two profiles: one for Residents and one for Assignees. Normally one profile manager will contain a set of user ID profiles that have something in common, such as being all in one department or being users of one application. The key is that the users defined in the profiles should all be needed on one set of target node subscribers. For example, you could have a department with Novell servers and UNIX machines and define all the user IDs within a profile manager so that all the users have consistent user IDs on each system type.

In our simple case, we followed the following sequence:

1. Create a Profile Manager called *ITSO_Users*, as shown in Figure 224 and Figure 225 on page 256.

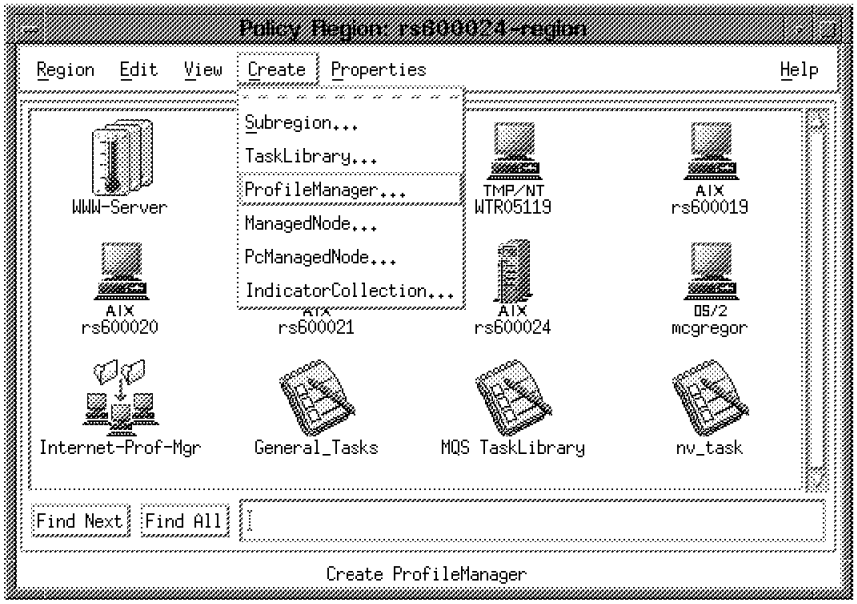


Figure 224. Create Profile Manager

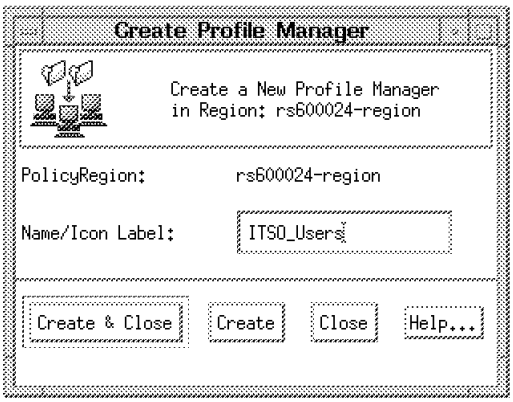


Figure 225. Create Profile Manager ITSO_Users

As Figure 224 on page 255 shows, you have to create the Profile Manager within the context of a Policy Region and when you have done so it will appear as a symbol in the Policy Region panel.

2. Open the ITSO_Users Profile Manager by double-clicking on the symbol in the Policy Region panel. This will display a new Profile Manager window, as shown in Figure 226.

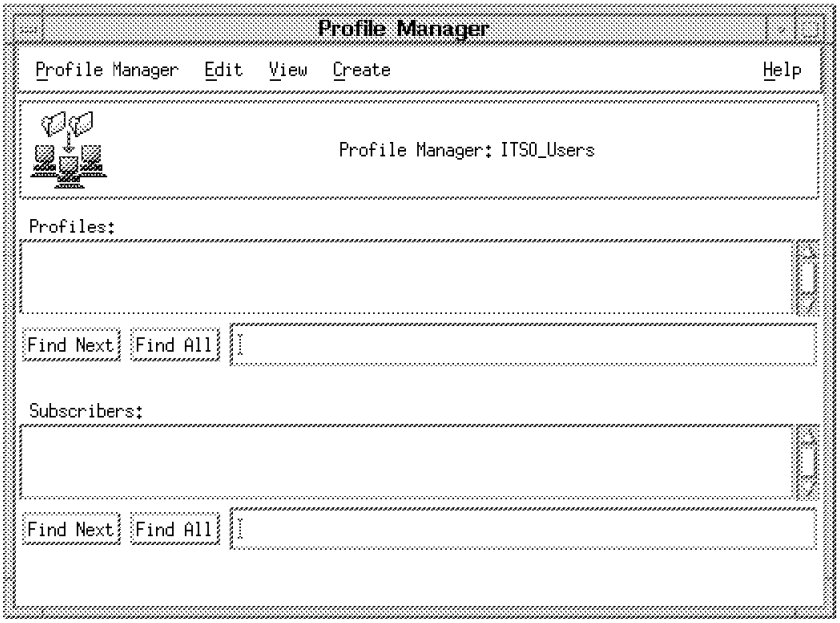


Figure 226. ITSO_Users with No Profiles

3. Next, select **Create** from the menu bar followed by **Profile**. The resulting dialog is shown in Figure 227 on page 257. Note that the list of possible profile types includes two Tivoli/Admin profiles, *GroupProfile* and *UserProfile*. If you do not see these profiles, you probably forgot to add the Admin resources to the containing Policy Region (refer to Figure 223 on page 254).

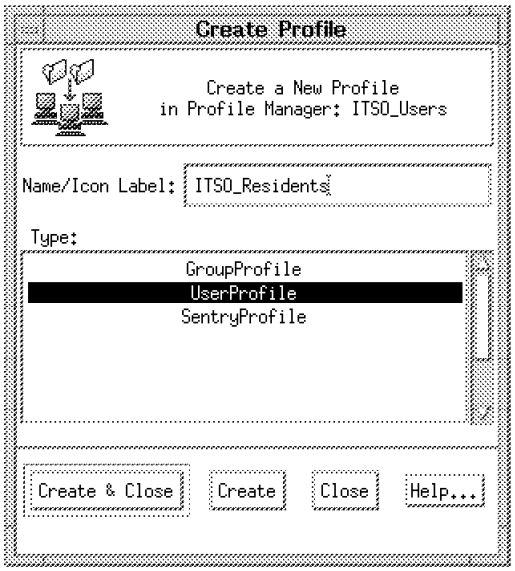


Figure 227. Create ITSO_Residents

Enter the name ITSO_Residents and select **UserProfile** from the Type list box. Then repeat for the ITSO_Assignees profile.

You will now see the two profiles represented by symbols in the Profile Manager panel, as shown in Figure 228.

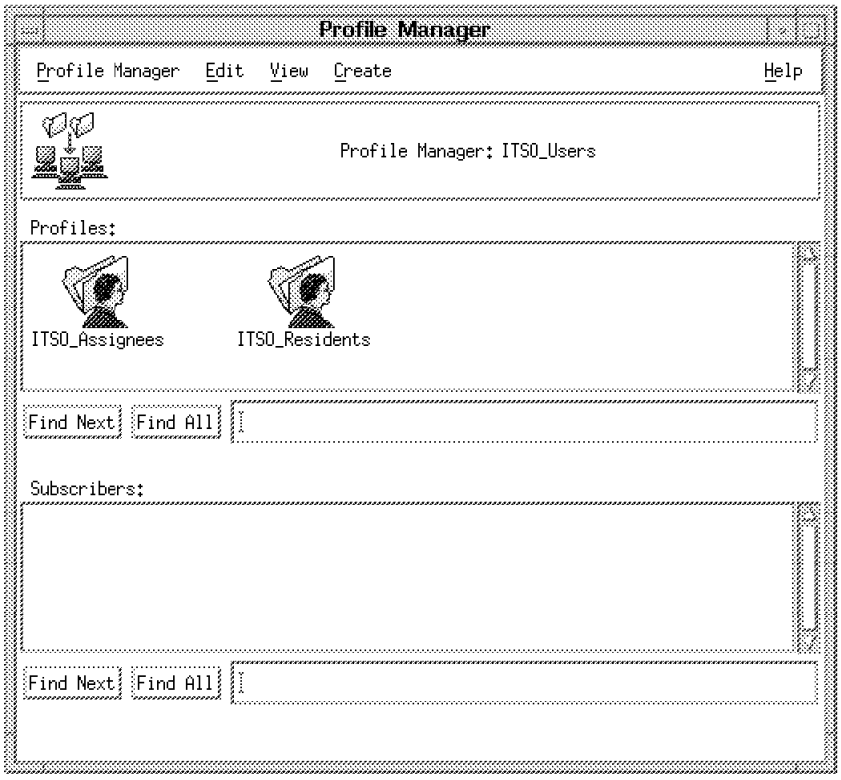


Figure 228. ITSO_Users with ITSO_Assignees and ITSO_Residents Profiles

8.2.3 User Profile Policies

By adding profiles in this way we have implicitly set some default policies for user ID creation. Before we start adding any records (users) to the profiles, we may need to review and change the default and validation policies for each user profile.

The *default* policy is a kind of template that can be used for every new user that is added. The *validation* policy is a way to check that values entered for a user comply with the policy rules.

To update the policies for the user ID profiles, do the following:

- Open the profile ITSO_Assignees by double-clicking on it. Select **Edit** and then **Default Policies** from the menu bar, as shown in Figure 229 and Figure 230 on page 259.

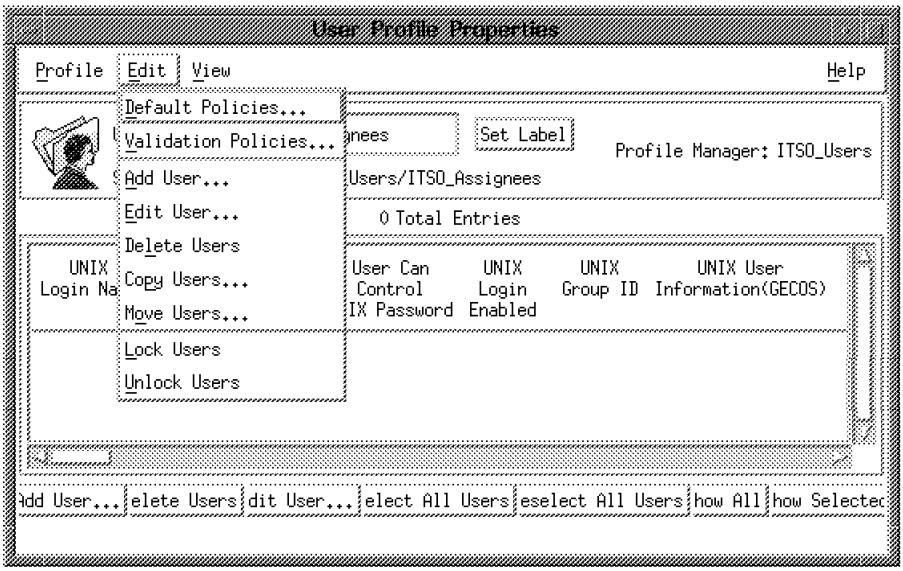


Figure 229. Editing Default Policies

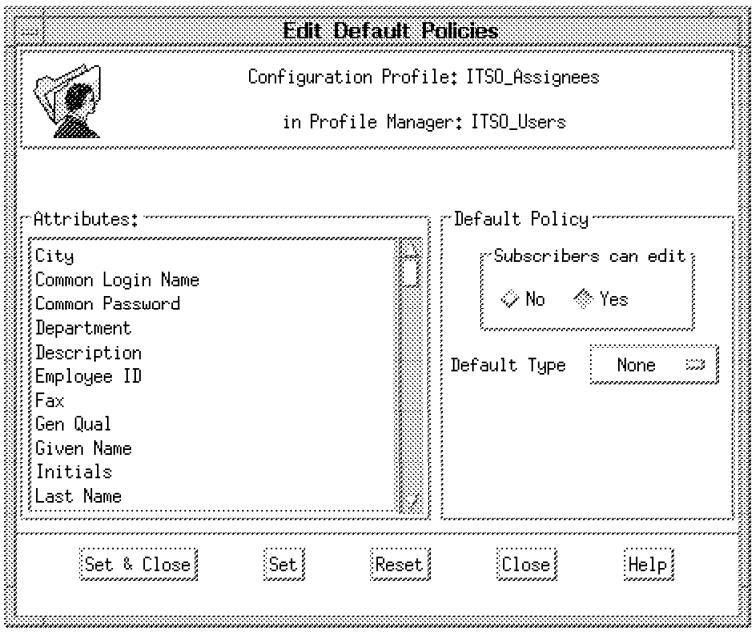


Figure 230. Edit Default Policies Window

Here you can see a list of the attributes that apply to a user. If you click on an attribute you will see the current default policy setting. The default policy contains the following information:

Subscribers can edit This defines whether a policy can be changed by a subscriber. This means that when a profile is distributed to a profile manager or to a managed node, you can apply different policies to fields in the user records before distributing them to their end points. Why would you want this capability? Typically it is provided to enable local administrators to customize the user information that a central adminstrator has provided.

Default Type This will be one of:

None This means that no default value will be supplied for this attribute when you add a new user.

Constant This will insert a fixed value into the field when you add a new user. For Constant type defaults, a Value field will appear when you select the attribute from the list, as shown in Figure 231 on page 260.

Script This means that a shell script will be executed when you create a new user. This script will provide the default value for the attribute. For Script type defaults, two additional fields will appear when you select the attribute from the list:

- Edit Script Arguments
- Edit Script Body

In our case we want to make a few changes to the default policies, as follows:

- Users in the ITSO_Assignees profile work in the same department and in the same location, so we can set default values for the *City* and *Department* attributes. Figure 231 on page 260 shows how we set the default for the City attribute.
- Users in the ITSO_Residents profile come from all around the world, so a default value for *City* and *Department* attributes will not apply.

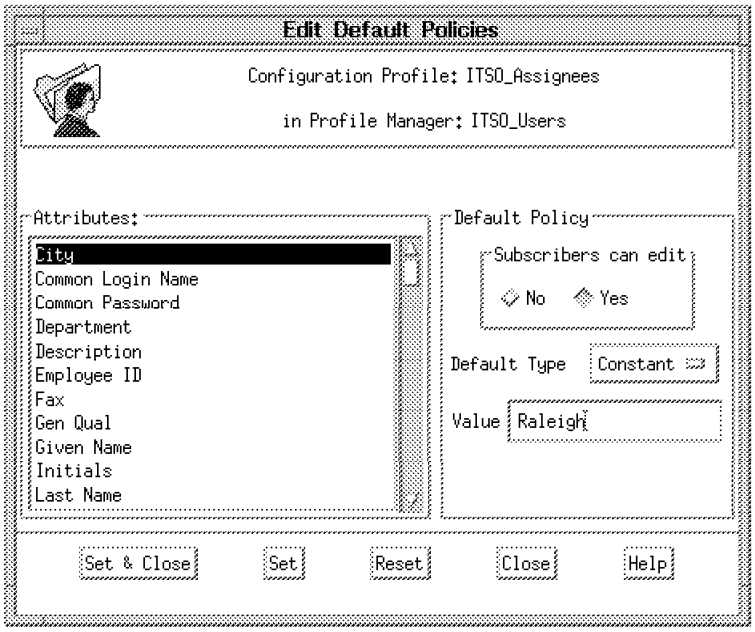


Figure 231. Enter a Value for Attribute City

We also wanted to modify the script associated with the policy for allocating the user ID number (the UID). The standard behavior is to set the UID to the first available number greater than 100. We decided that assignees and residents should have their UIDs allocated from different blocks of numbers (assignees' UIDs starting at 500 and residents at 600). To achieve this, we performed the following steps:

1. In Profile ITSO_Assignees, select the attribute **UNIX User ID**.
2. The default type for this attribute is Script, so select **Edit Script Body**.

The script is now displayed and can be edited, as shown in Figure 232 on page 261.

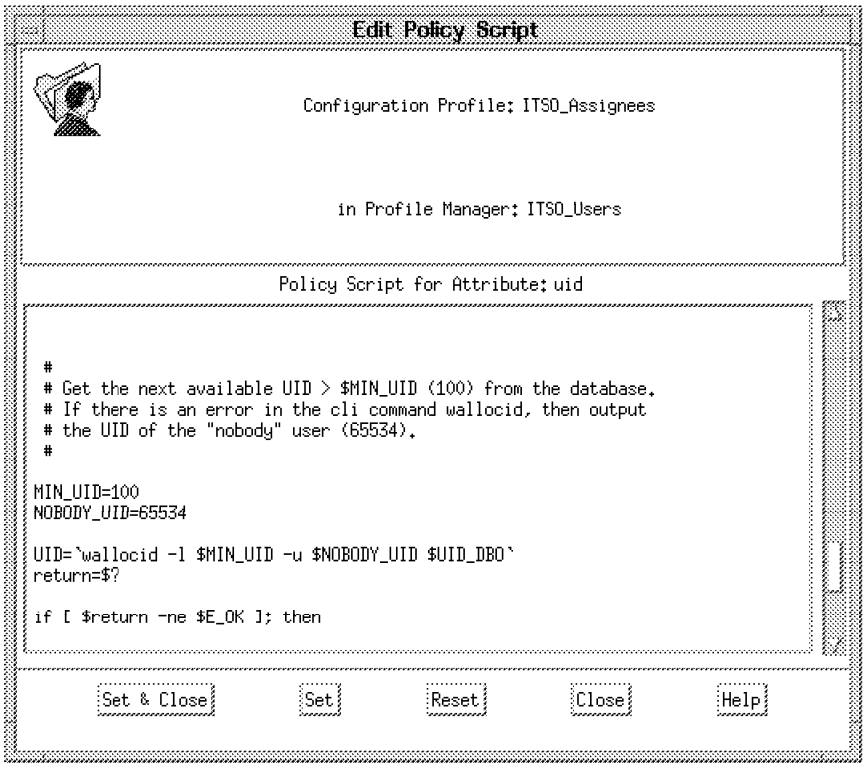


Figure 232. Edit the Policy Script for Attribute UID

The default script is in fact a very simple shell script. To alter the behavior we simply changed the line MIN_UID=100 to MIN_UID=500.

Changing the default policy in this way only alters the initial values that are filled in when you create a new user. To ensure that the new default policy is enforced, we must edit the validation policy for the UNIX User ID attribute to check that the number is in the correct range. For example, if you add a user and enter a value for this field before clicking on Generate Defaults, or change the default value it generates, the default policy will not be enforced. Setting the validation policy will prevent incorrect values from being passed through.

To set a validation policy, select **Edit** and then **Validation Policies** from the User Profile window menu bar. Edit the validation script for UID in the same way as the default policy script.

8.2.4 Managing User Profiles

At this stage we still have no records (users) defined in the User Profile. We can populate the user profiles with existing user information from specified endpoints. Populating the profile in this way does not enforce the default policy attributes. The values put in the profile are those found in the system files.

In our example, users for the ITSO_Assignees and ITSO_Residents profiles are already defined on managed node rs600024, so we can use that information to populate the profiles.

To populate a profile from the Profile Manager window, do the following:

1. Open **Profile** by double-clicking on it and select **Populate** (see Figure 233 on page 262).

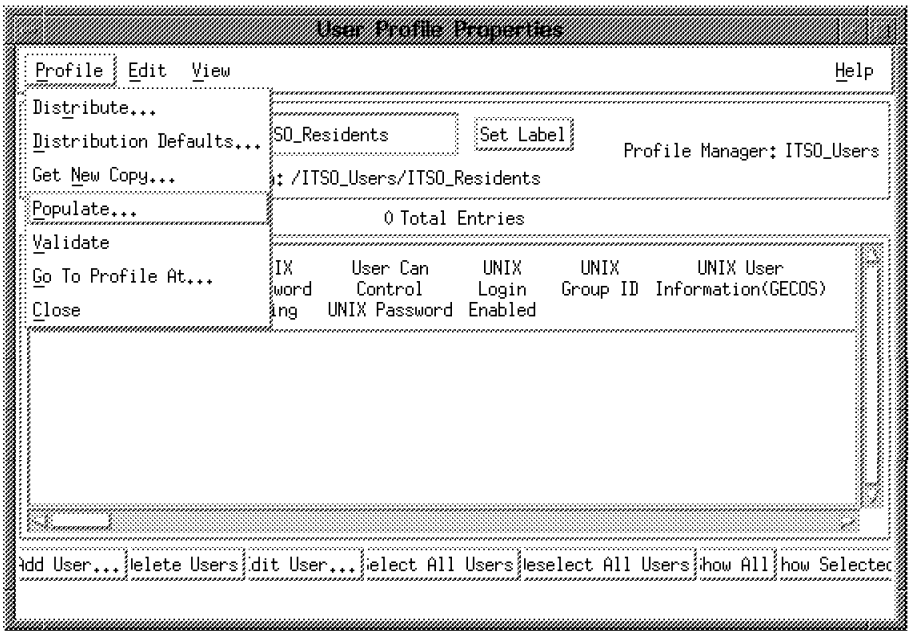


Figure 233. Populate the Profile

2. Select managed node **rs600024** to get the records (see Figure 234).

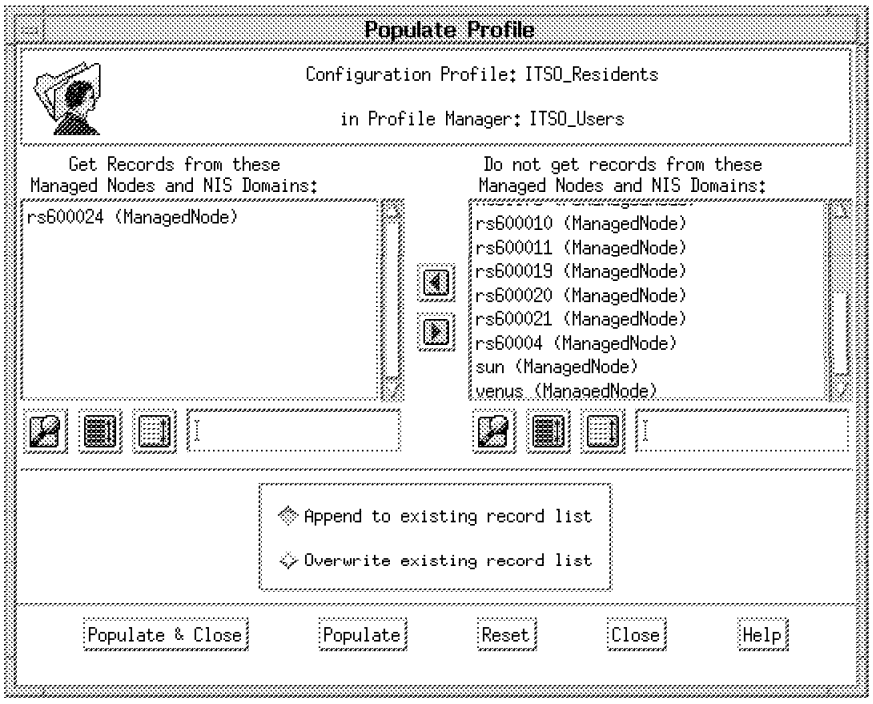


Figure 234. Get Records from Managed Nodes

The validation policy will be checked as the records are created and warning messages will be issued, as shown in Figure 235 on page 263. Records that fail the validation policy will not be added to the user profile.

In this case we have set the validation policy for the **UNIX User ID** to check for a minimum value of 500. All our users on Managed Node **rs600024** have UIDs below 500, so will not be added to the user profile.

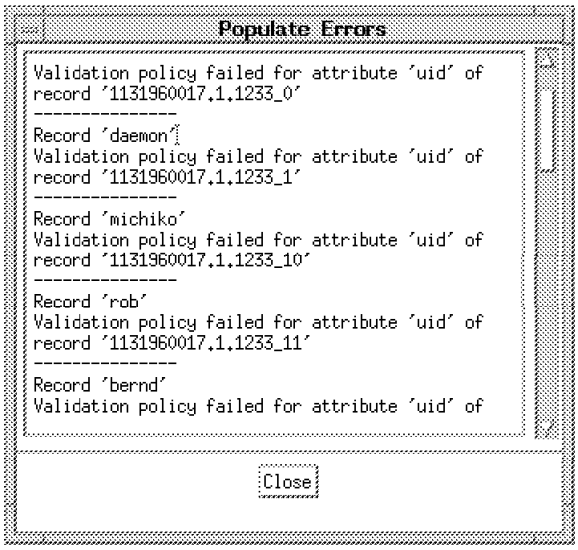


Figure 235. Populate Errors

However we still wanted these records to be added, but also needed to enforce the policy for any new users added in the future. There are two ways to achieve this:

a. Populate the user profile before changing the validation policy.

b. Disable the **Validation Policy** from the **Edit Validation Policies** window before populating the profile, then enable it again.

We chose the first option.

3. All the users on rs600024 that now pass the default Validation Policy will be added to the profile, as shown in Figure 236 on page 264.

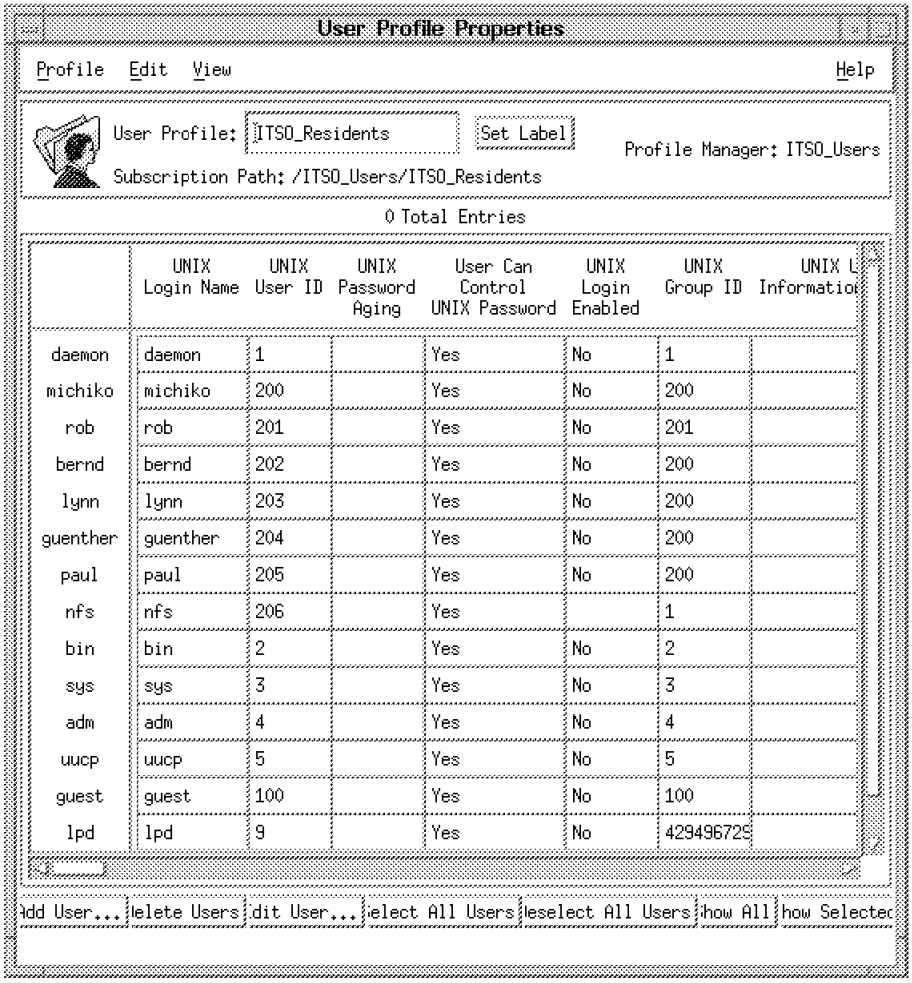


Figure 236. ITSO_Residents Populated with Records from rs600024

Note that the default validation policy for the **UNIX User ID** checks for a UID value greater than 0. This means that user **root (0)** and **nobody (-2)** fail the test and are not added. So if you want **root** and **nobody** to be included, you will need to disable the validation policy or change it to include UIDs with values of less than 0.

Now we have a populated profile, but in fact we want only the *resident* users to be recorded in this profile. We therefore need to remove those we do not want:

1. Select multiple users you want to delete by holding down the Ctrl key and selecting the users with the left mouse button.

Attention

When the user profile is populated using the GUI or with the `wpopusrs` command, or you add a user manually to the profile, Tivoli/Admin is not yet really managing those users. It is only after you distribute new or edited records that they become *managed*.

Deleting users from the profile before any distribution has been done is safe, but if you delete a user from a profile when Tivoli knows about it, the user will actually be deleted from the endpoint system files when you next distribute the profile.

See 8.2.9, “Deleting User Records” on page 286 for more details.

2. Select **Delete Users** to delete the unwanted users (see Figure 237).

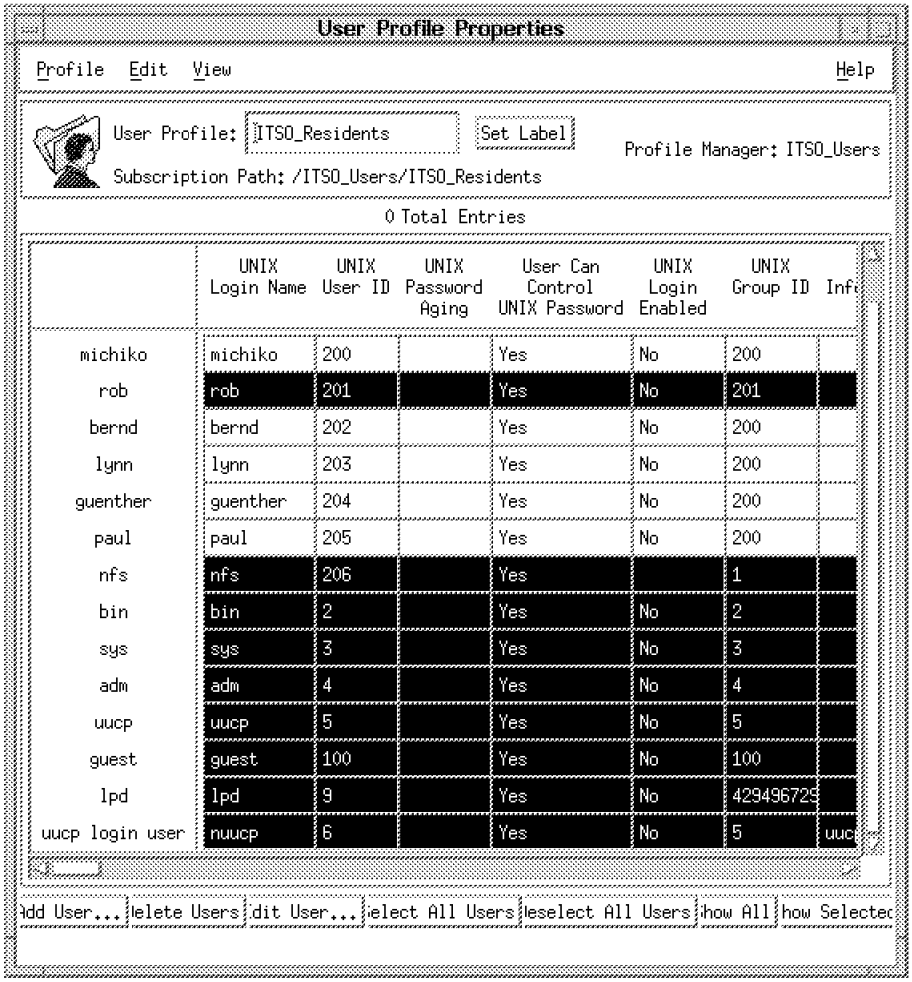


Figure 237. Select Users to Delete

3. You will be prompted to respond whether the home directory of the users should be deleted or not. At this stage, this does not have much meaning because the profile has not yet been distributed. Thus, the user IDs will not actually be deleted.

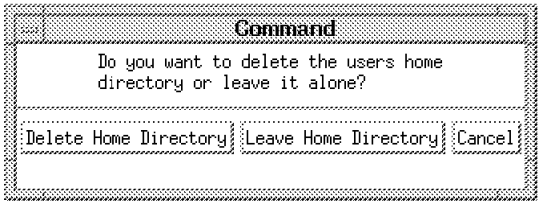


Figure 238. Delete or Leave Home Directory

4. Select **Leave Home Directory**. You will be left with the users you want, as shown in Figure 239.

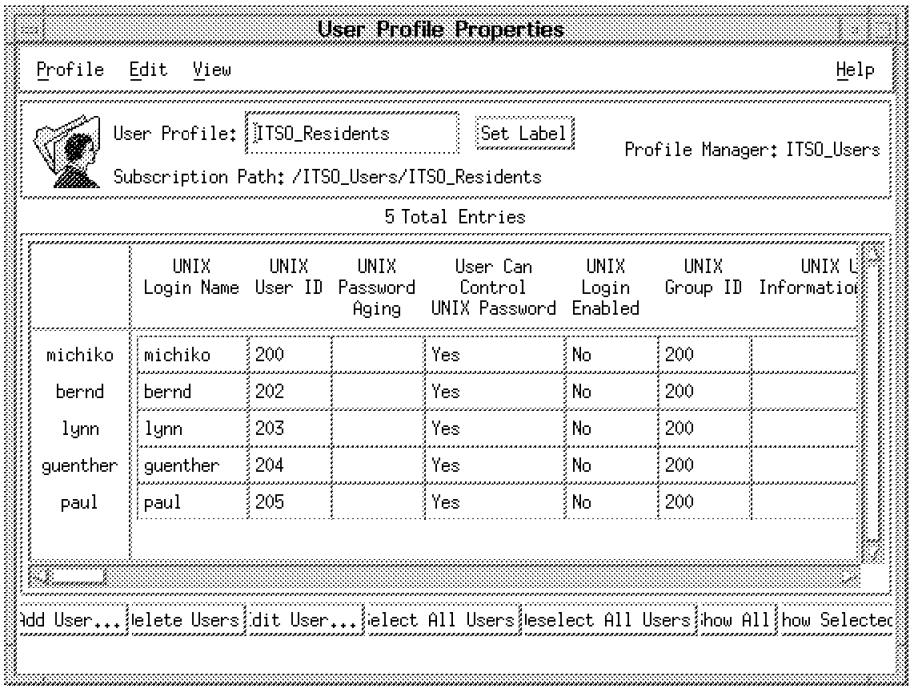


Figure 239. ITSO_Residents Records

Command Line Option

Alternatively, you can populate the profile using the `wpopusrs` command and specifying a filename that contains a list of the users you want, for example:

```
wpopusrs -f /tmp/listofusers -o -l @ManagedNode:rs600024  \
         @UserProfile:ITSO_Assignees
```

(where `@ManagedNode:rs600024` is the source for the user information, `@UserProfile:ITSO_Assignees` is the profile to be populated and `/tmp/listofusers` contains a list of user names). In our case we reduced the list of assignees to just one user record (see Figure 240 on page 267).

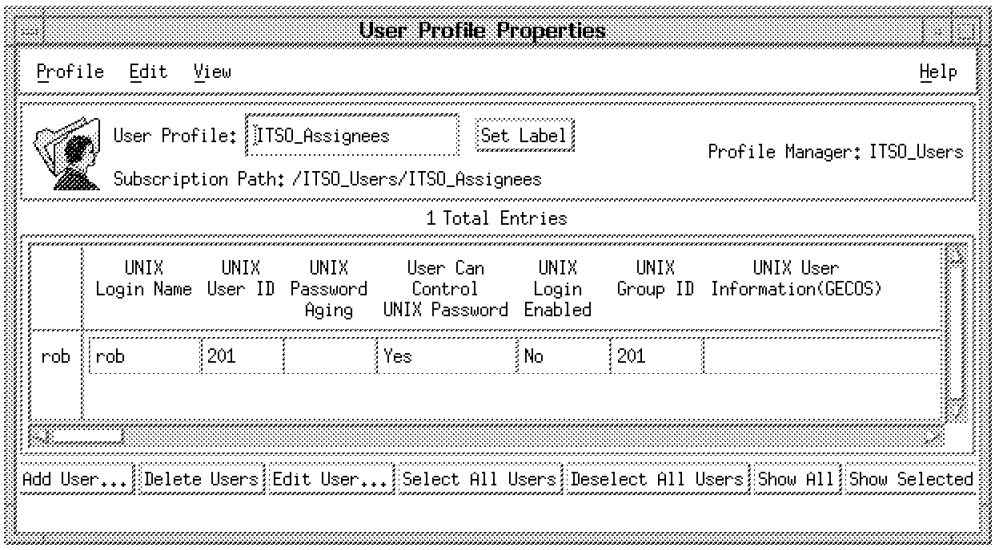


Figure 240. ITSO_Assignees Record After wpopusrs Command

8.2.5 Setting Subscribers

Having created the user profiles (and thus defined the user details within TME), the next step is to distribute them. For this we have to define subscribers to the User Profiles. There are two levels at which subscribers can be set:

Profile Manager This level determines to which endpoints and profile managers TME will attempt to distribute the contents of a profile.

Record (user) Subscribers are specified for each record within a profile to determine which machines will actually have their system files updated.

When a User Profile is distributed, all of the user records are sent to the specified subscribers, but the system files will only be updated where the record itself has a subscriber defined.

The following example shows how to distribute to a Profile Manager as well as to an endpoint:

1. Create a Profile Manager called Test User Profile Distribution (see Figure 241 on page 268).

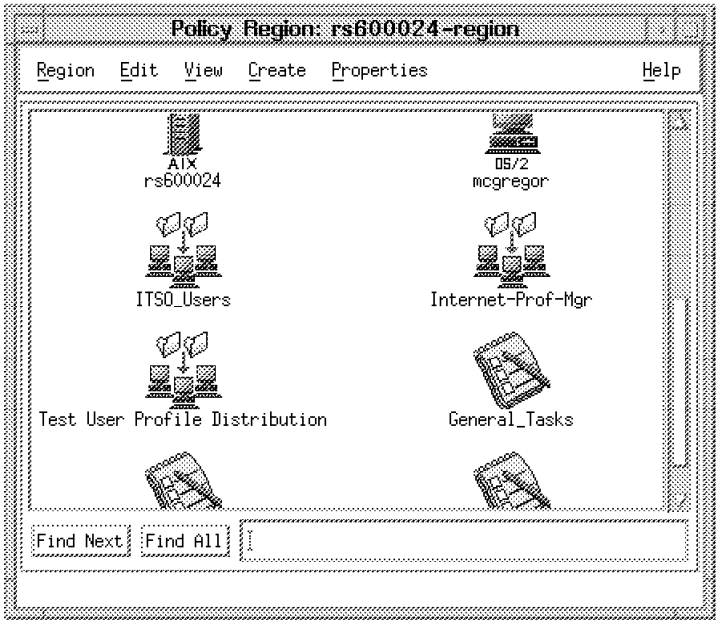


Figure 241. Create Profile Manager Test User Profile Distribution

2. Add subscribers to this Profile Manager. In this example we select nodes rs600021 and rs600020 as subscribers (see Figure 242 and Figure 243 on page 269).

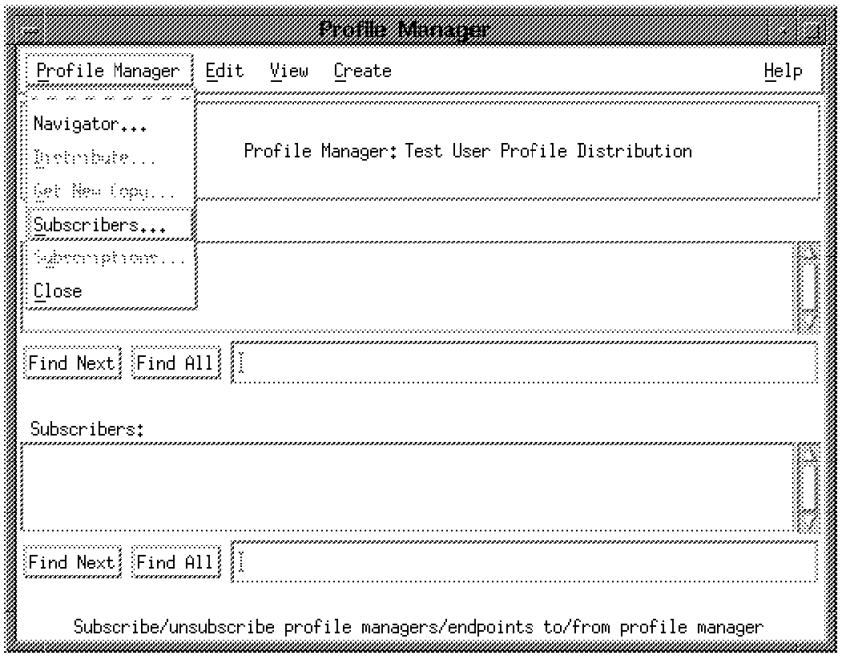


Figure 242. Add Subscribers to Profile Manager

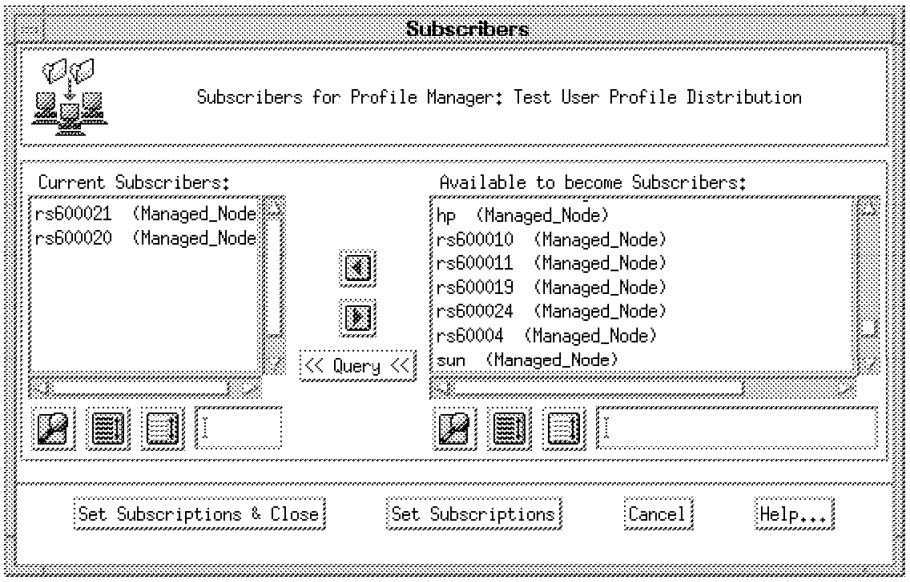


Figure 243. Set Current Subscribers

3. Open the ITSO_Users profile manager and select **Subscribers** from the **Profile Manager** menu bar. We selected two subscribers here (see Figure 244). One is the profile manager Test User Profile Distribution which we have just created (thereby creating a hierarchy of profile managers), and the other is the single managed node rs600019.

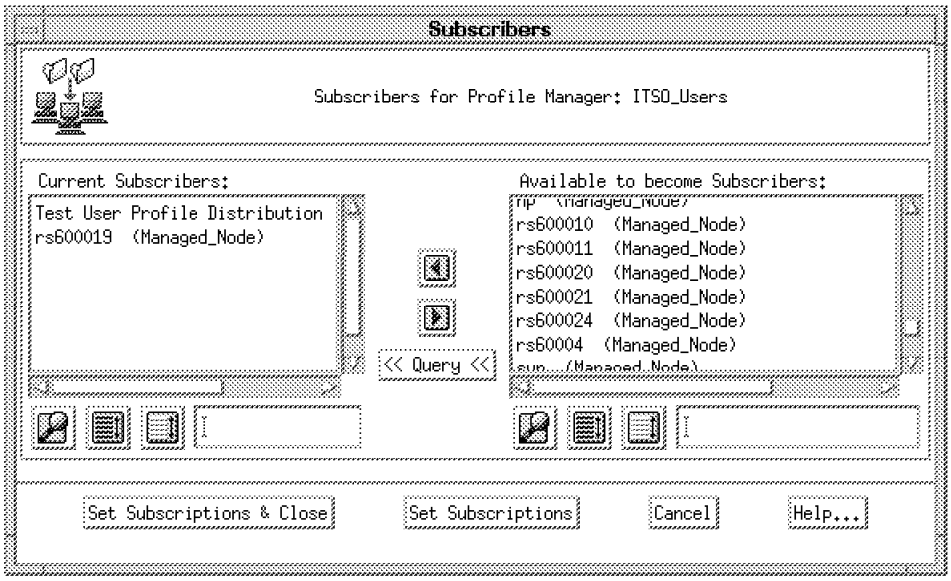


Figure 244. Add Subscribers to ITSO_Users Profile Manager

The contents of the profiles can now be distributed to the subscribers. The system files will not be updated unless the individual records also have the subscribers set. In fact, all the records that now populate the user profiles will *automatically* have the same subscribers added that subscribe to the ITSO_Users Profile Manager. This means that you can at this point edit each record to change the list of subscribers, or just leave them alone if you are happy with the distribution scheme inherited from the Profile Manager.

For example, to remove all subscribers from user *rob* in Profile ITSO_Assignees you would do the following:

1. Select rob from the list of users in ITSO_Assignees and select **Edit User** (see Figure 245).

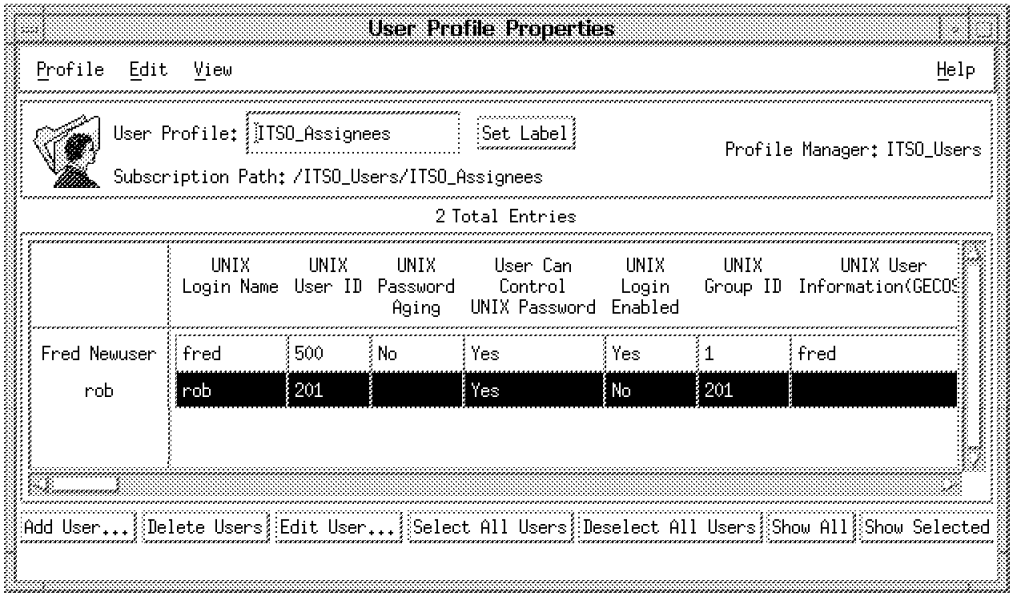


Figure 245. Edit Record in User Profile

2. Select **Subscribers** from the category list and move all Current Subscribers to Available Subscribers by clicking the right arrow button (see Figure 246 on page 271).

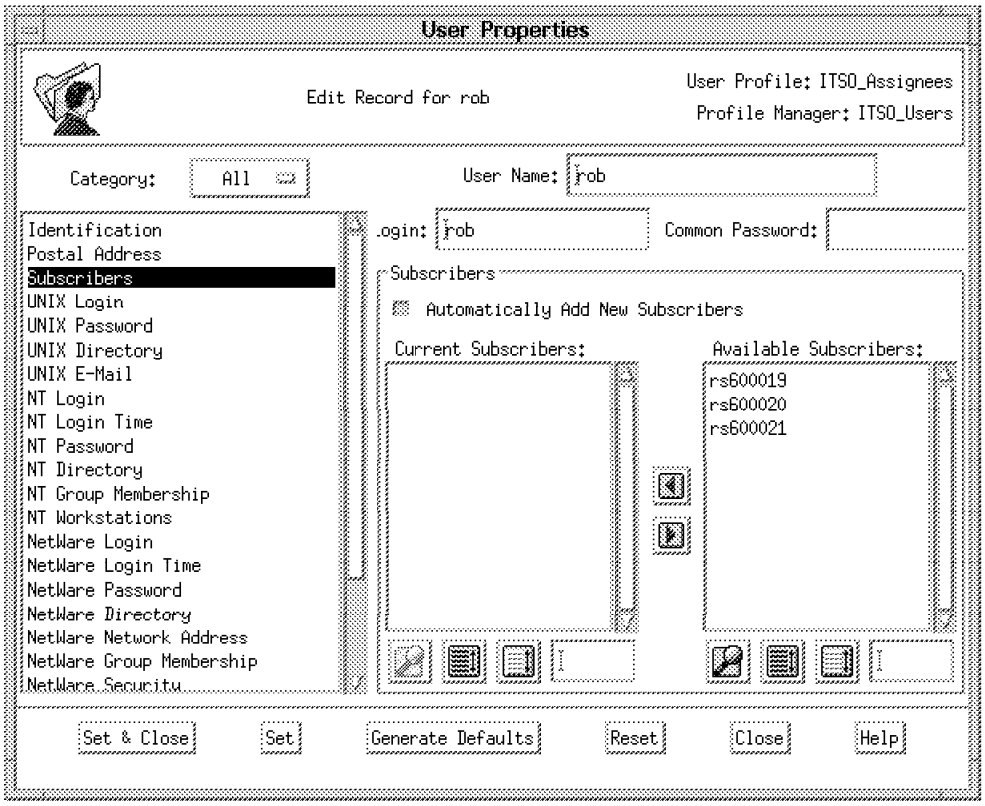


Figure 246. Remove all Current Subscribers

3. Click on **Set & Close**.

Now when ITSO_Residents is distributed, no system files will be updated for user rob.

8.2.6 Create a New User

Before we test the distribution, we will create a new user in the ITSO_Assignees profile. We will limit its subscribers to rs600021 and rs600019.

1. Open ITSO_Assignees and select **Add User** from the **Profile** menu bar entry. The resulting dialog is shown in Figure 247 on page 272. Notice that in the one record you define all possible user attributes: general, UNIX, NT and NetWare. This means that you can define the user once, and if that user needs access to systems of multiple types you can administer the access in a centralized, consistent way.

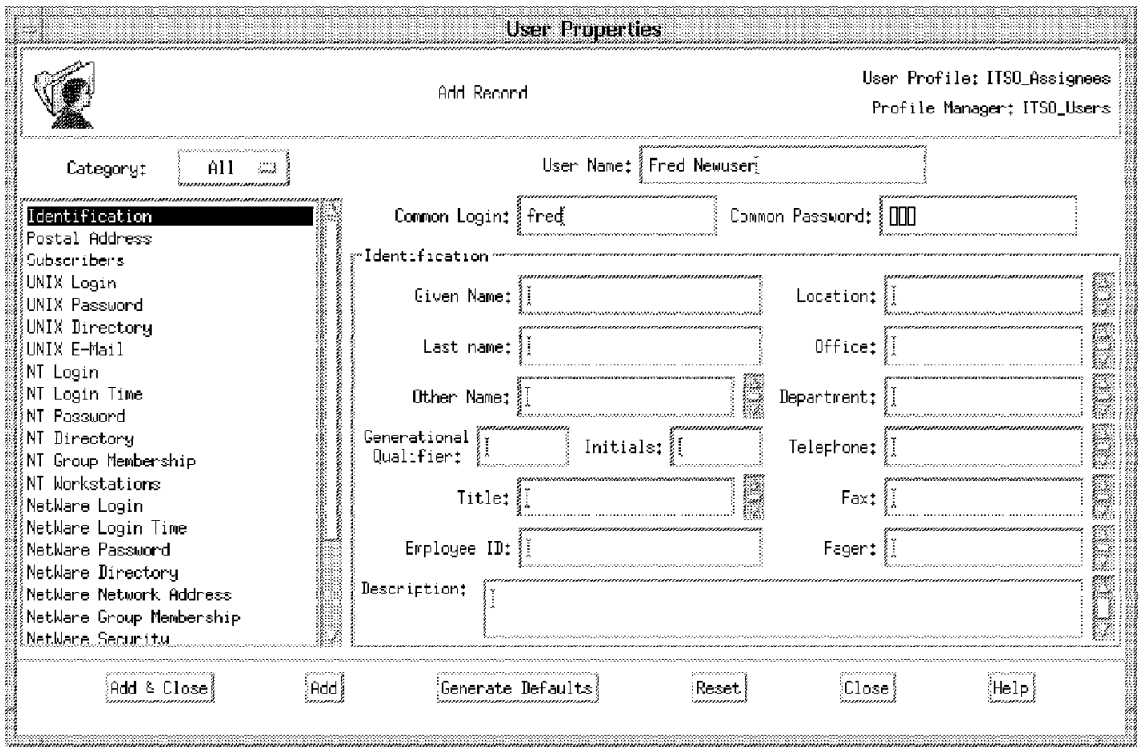


Figure 247. Fill in User Name, Common Login and Common Password

2. Having entered the basic user information (name, common login name and common password) you should then click on **Generate Defaults** (see Figure 248).

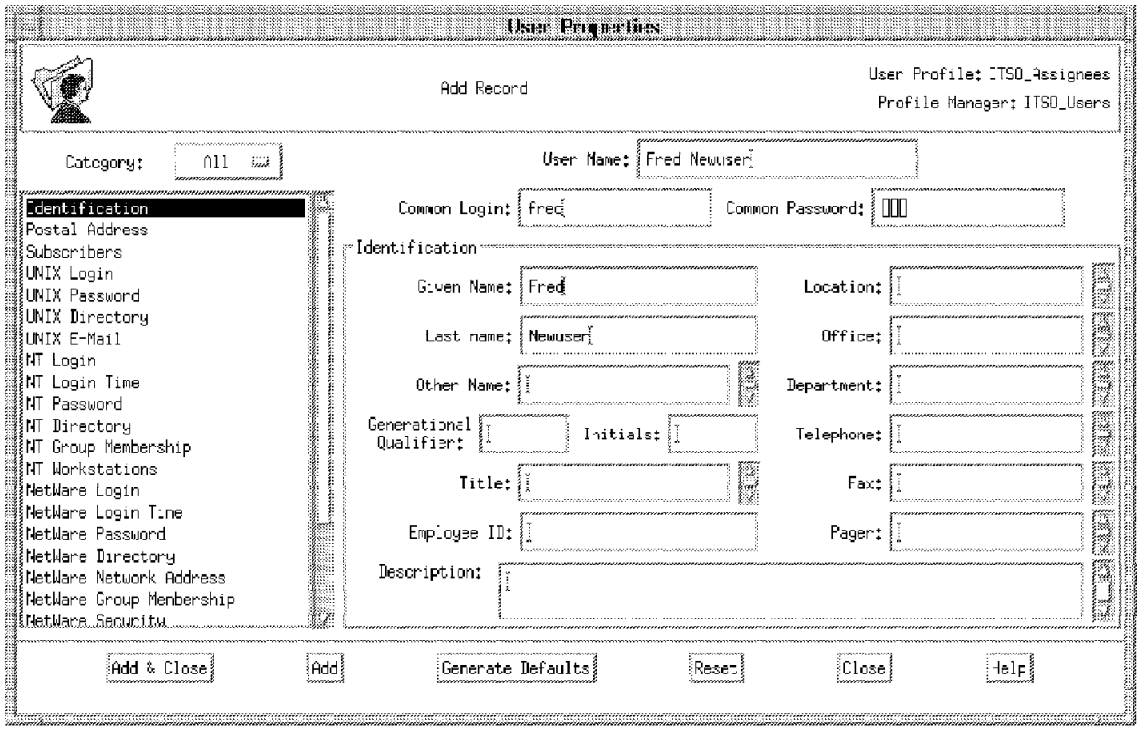


Figure 248. Default Values are Filled in with Generate Defaults

You can see from Figure 248 that the Given Name and Last Name values have have been filled in by the default script supplied for those attributes.

In fact, the **User Name** is the only field that must be filled in. The **Common Login** and **Common Password** fields can be generated from defaults.

3. If you select a category from the category list, the dialog will display panels of attributes specific to that category. Some of these have default values, but you can override them manually or supply values for attributes that do not have defaults.

 You can limit the number of categories displayed by selecting an operating system type from the Category pop-up menu, as shown in Figure 249.

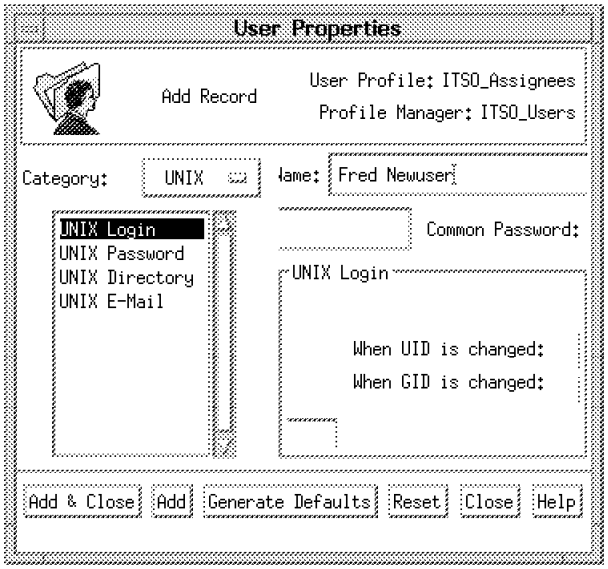


Figure 249. Limit the Category List

4. If you select **Subscribers** from the category list you will see that the subscribers to the Profile Manager have automatically been replicated for this user. In fact, as Figure 250 on page 274 shows, the Profile Manager subscriber (Test User Distribution) to ITSO_Users has been expanded into the individual endpoint subscribers.

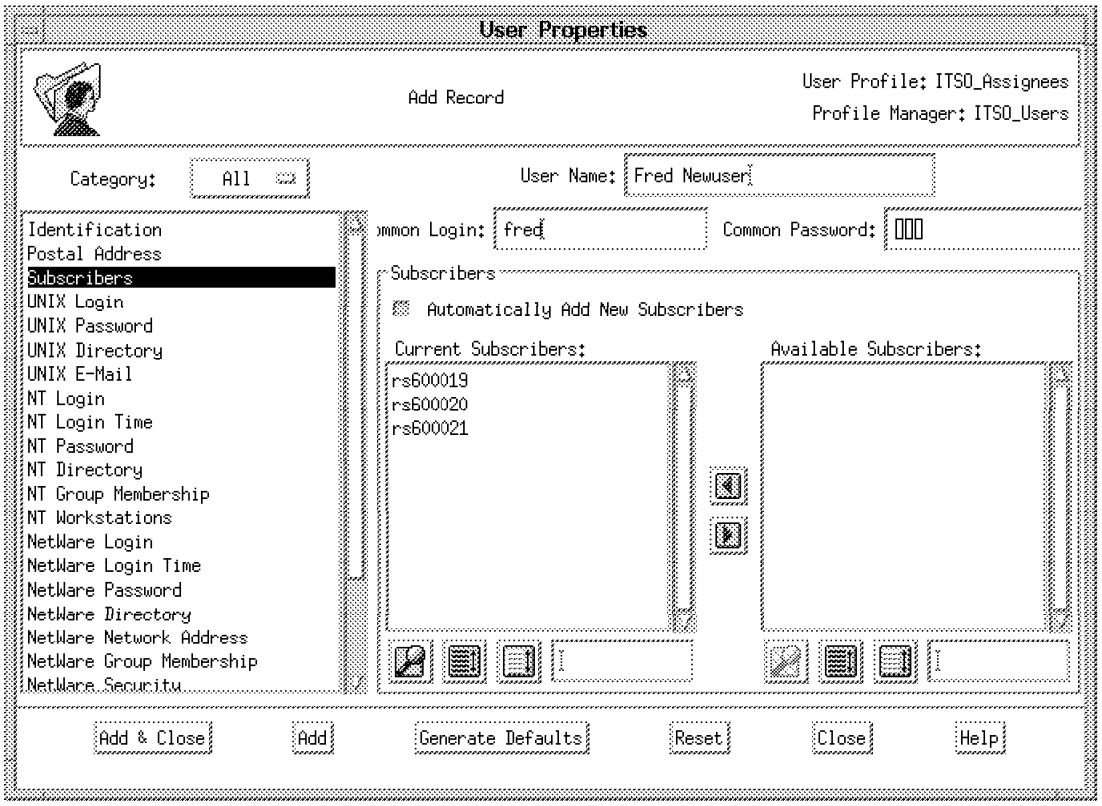


Figure 250. Current Subscribers Category

5. We can choose to remove subscribers from this record if we do not want to distribute this record to all the subscribers to the profile manager. In this case, we remove rs600020 from the list of subscribers and then click on **Automatically Add New Subscribers** to prevent new subscribers that are added to the Profile Manager from being automatically added to this record (see Figure 251 on page 275).

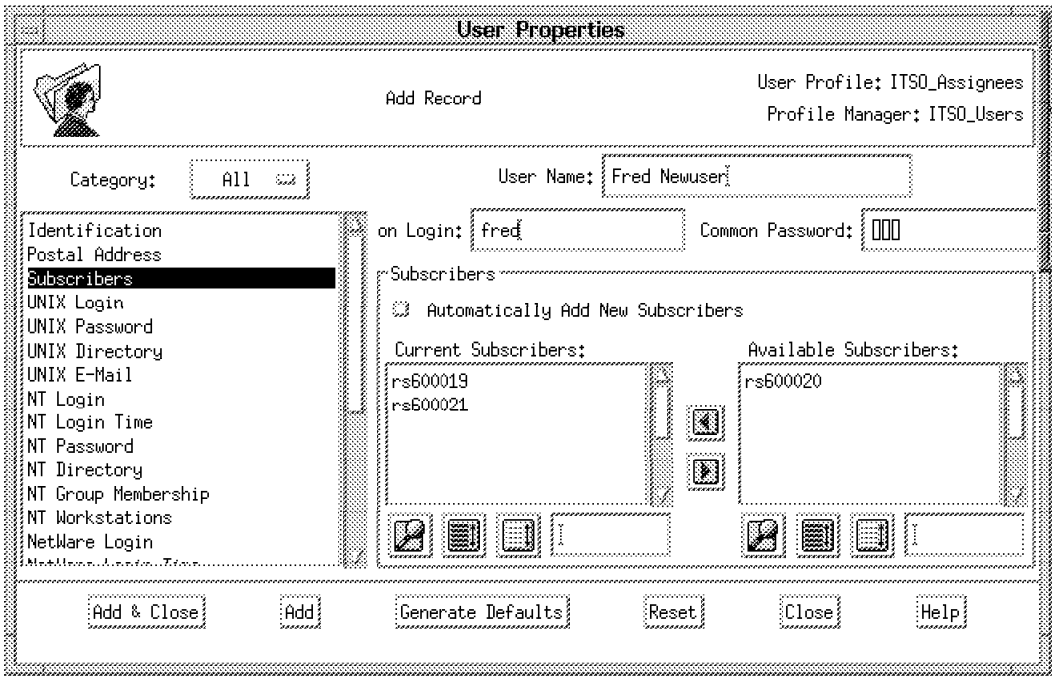


Figure 251. Change Current Subscribers and Automatic Subscription Behavior

The net result of this change is that the contents of the record will be distributed to all subscribers to the ITSO_Users Profile Manager, but the system files will only be updated for this user ID on rs600019 and rs600021.

6. Click on **Add & Close** to add the user to the profile, as shown in Figure 252. Note that the UNIX User ID has been set at 500, following the rules in the Default Policy for this field that we set up earlier.

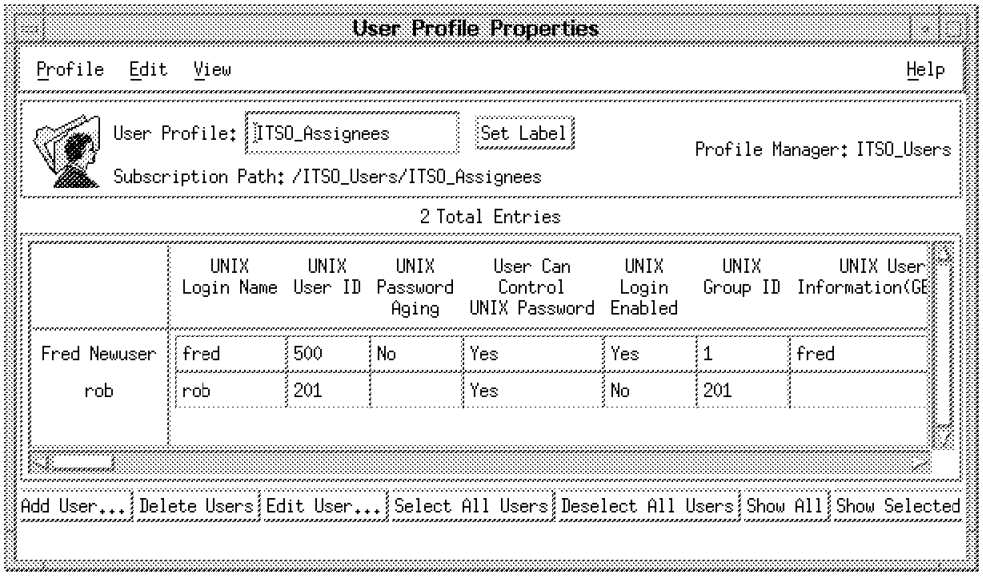


Figure 252. New User Added to Profile ITSO_Assignees

8.2.7 Distribute User Profiles

Attention

We found there was nothing to stop us from creating the same user in more than one user profile and distributing the record to the same subscribers.

We created a user in one profile with UID=501 so that the user was also created at the endpoint. We then created the same user in a second User profile with UID=103. After distribution to the same endpoint, the system files were changed to show UID=103. This could get very confusing and even dangerous, so be careful.

Note: If you have the same user in more than one profile, do not distribute the profile record to the same subscribers.

Careful planning and design of your profile managers and user profiles should eliminate this problem.

We will now test the distribution of the new user we have just created. As we already removed subscribers from user rob, only user fred will be updated on its subscribing systems.

Before we distribute the user profile, we can set distribution defaults, as follows:

1. Select the **Distribution Defaults** option from the Profile menu item in the ITSO_Assignees User Profile (see Figure 253).

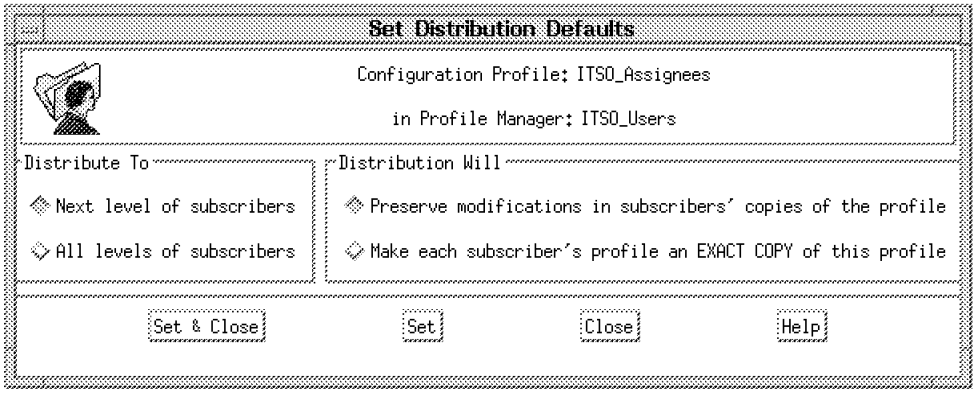


Figure 253. Setting Distribution Defaults

Separate distribution defaults can be set up for each profile. These defaults will apply when the distribution is initiated from the Profile Manager window or from the User Profile window. The distribution defaults are as follows:

Distribute To Next level of subscribers

Distribution Will Preserve modifications in subscribers' copies of the profile

For this example we will perform the distribution first using these default settings.

2. Select **Distribute** from the **Profile** menu item in the ITSO_Assignees User Profile, as shown in Figure 254 on page 277.

 You can see that you have another opportunity here to change the distribution values. If the distribution is initiated from the Profile Manager it will use the defaults set for each user profile. The subscribers listed under

Distribute to These Subscribers are those we set up earlier to subscribe to the Profile Manager.

As we have specified, the distribution should go to the next level of subscribers. No system files will be updated at any of the endpoints.

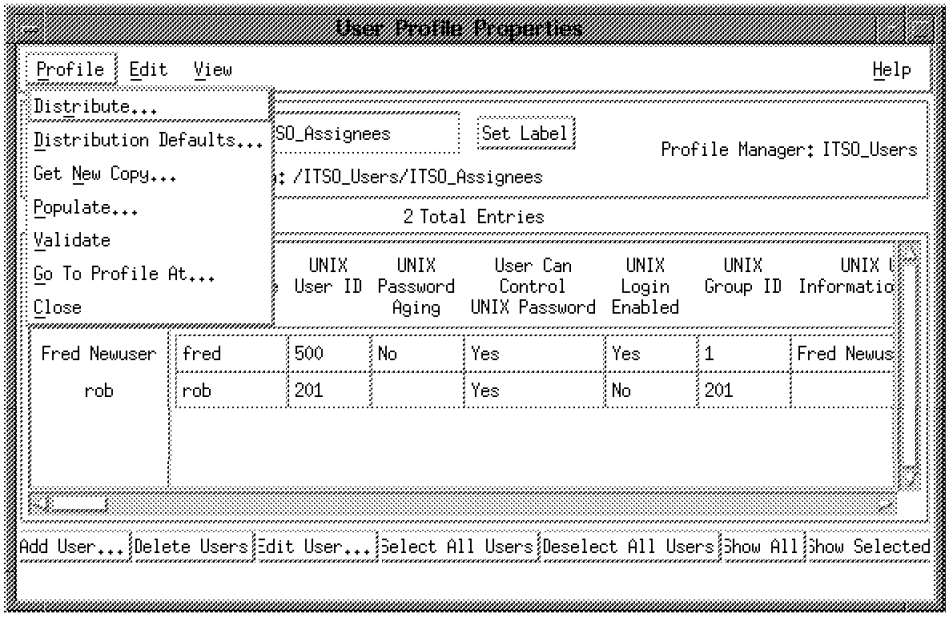


Figure 254. Distributing the Profile

Once we distribute the profile in this way, the result is that the subscribing profile manager Test User Profile Distribution and the Managed Node rs600019 will receive a copy of the ITSO_Assignees profile. We can see what has been received by opening the Profile Manager icon (either in Policy Region or under Subscribers of the ITSO_Users Profile Manager). Figure 255 on page 278 and Figure 256 on page 279 illustrate this.

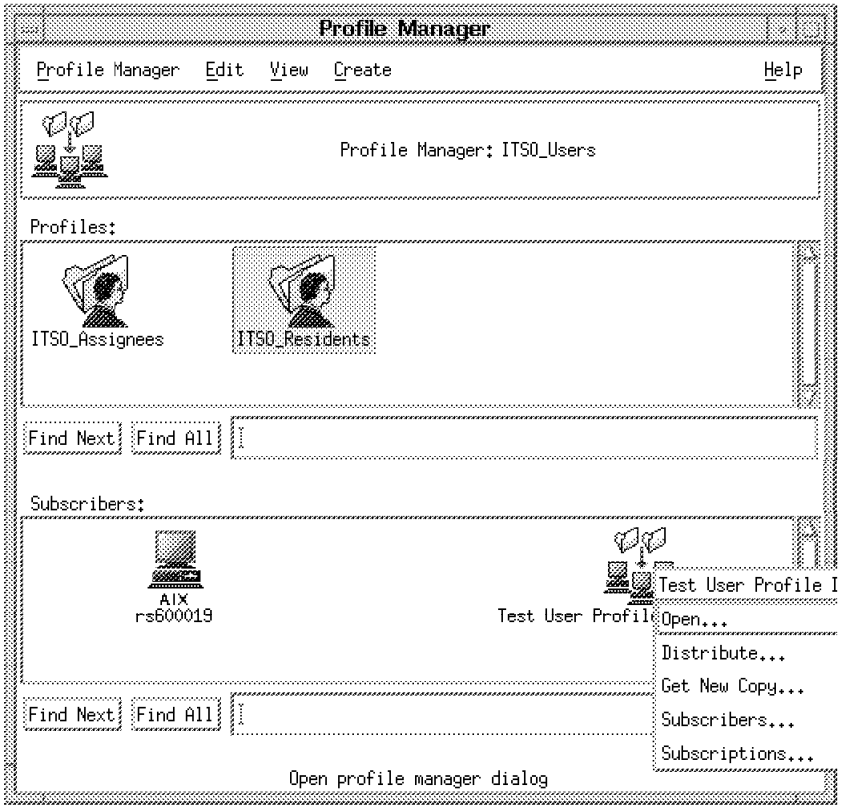


Figure 255. Open Subscriber Icon

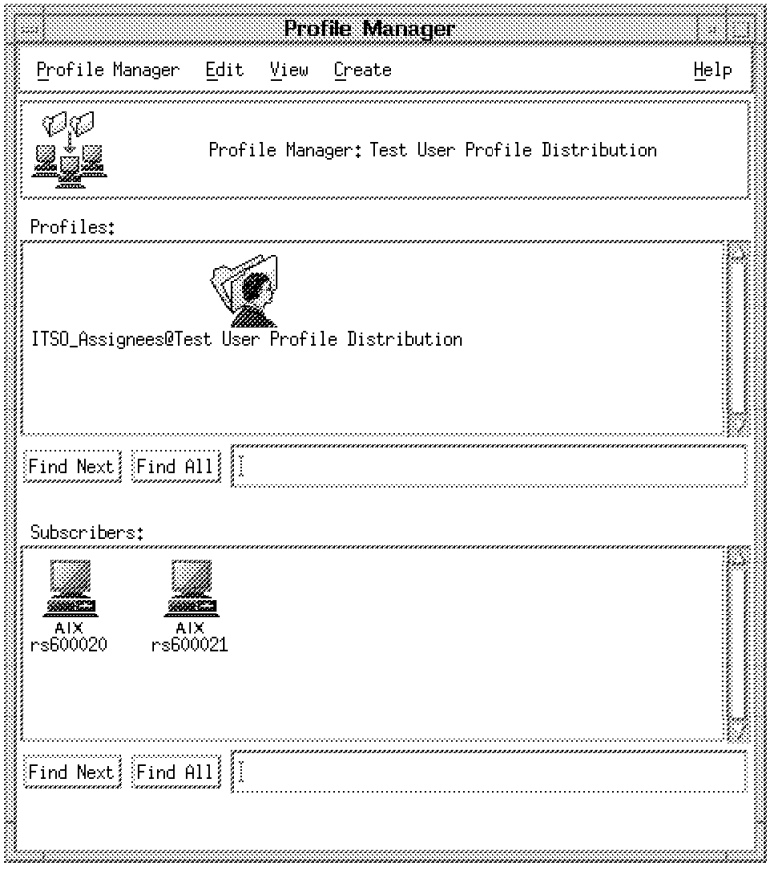


Figure 256. Subscriber's Copy of ITSO_Assignees

You can double-click on **ITSO_Assignees@Test User Profile Distribution** and make changes before continuing the distribution. In practice, this Profile Manager could be under the control of another administrator who will make the changes.

As an example, we created an administator called *User_adminstrator* whose only available resource is the Test User Profile Distribution Profile Manager. Figure 257 on page 280 and Figure 258 on page 281 show the desktop for this new administrator, and the window he sees when he opens the ITSO_Assignees@Test User Profile Distribution profile.

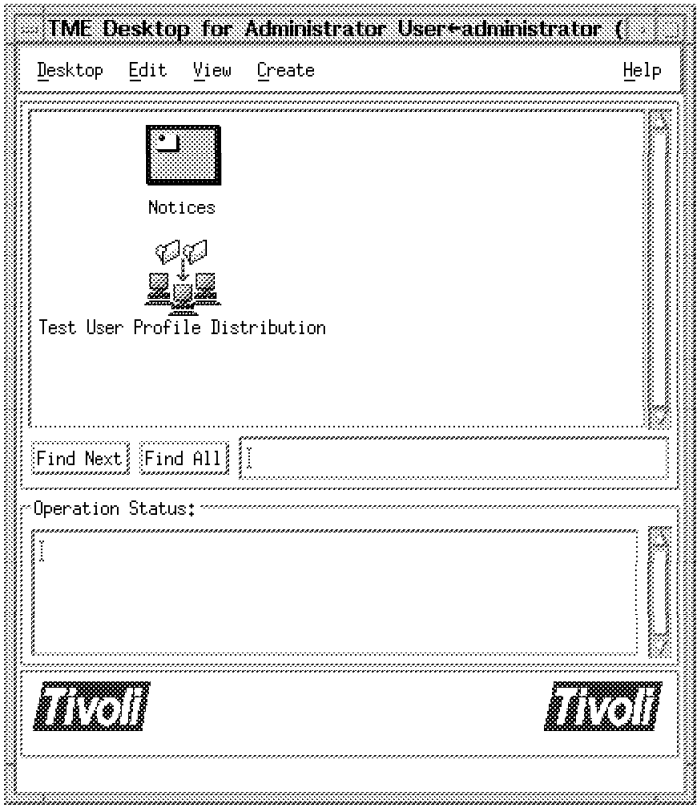


Figure 257. Resources Managed by Adminstrator User_admininistrator

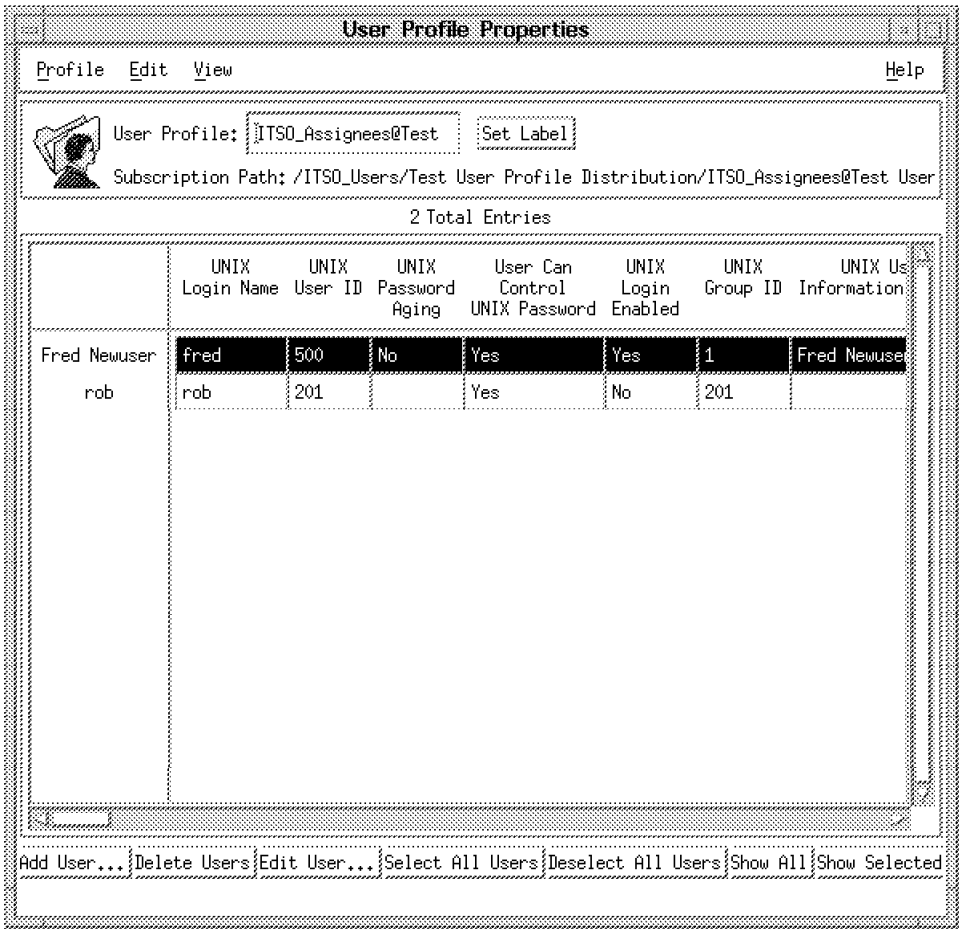


Figure 258. ITSO_Assignees@Test User Profile Distribution Profile

The administrator can make changes to these user profiles. As an example, Figure 259 on page 282 shows the result of overriding the UID value for the user fred.

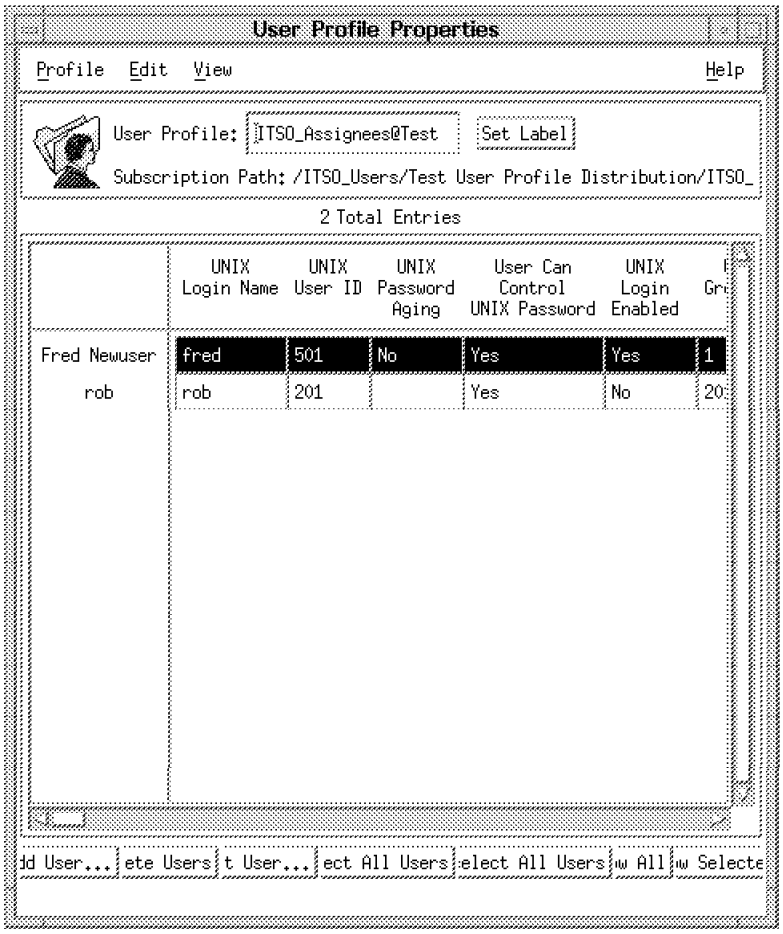


Figure 259. User Record has Been Changed

If we now distribute ITSO_Assignees@Test User Profile Distribution from User_administrator to **All levels of subscribers**, the system files on rs600021 will be updated. That is, the user fred will be created with UID=501. The system files on rs600020 will not be updated because we have removed that endpoint from fred's subscribers, although the profile itself will still be distributed to the managed node. See Figure 260 on page 283.

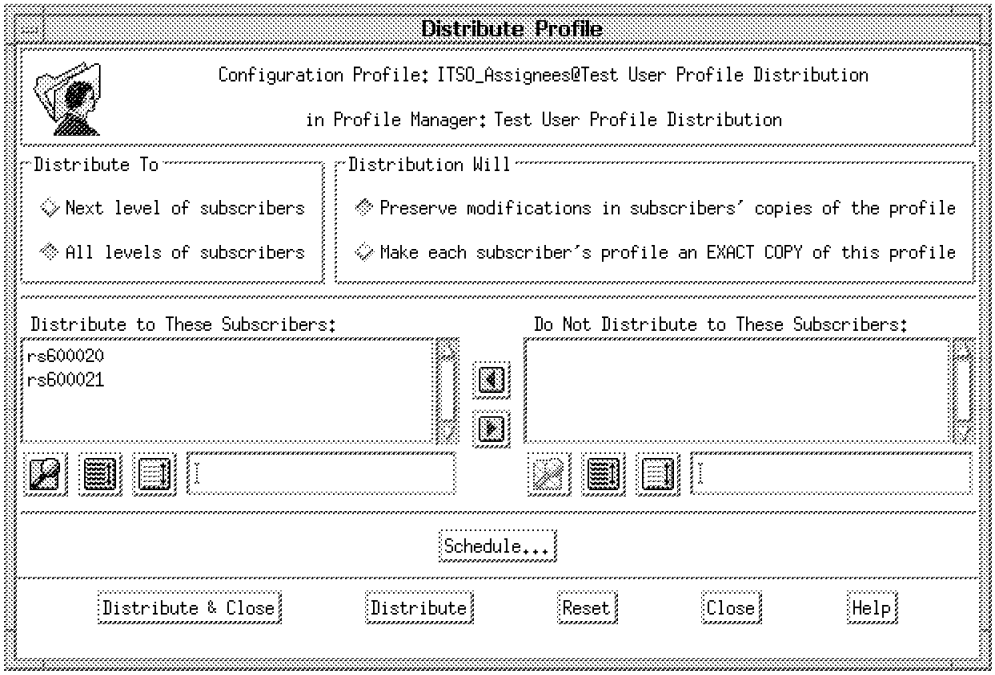


Figure 260. Distribute to All Levels of Subscribers

EXACT COPY Distribution

Do not use EXACT COPY distribution for this profile. This option completely replaces the endpoints system files to match *exactly* what you have in the profile. So where you have a profile that contains a subset of users, in this case just *rob* and *fred*, an EXACT COPY will leave /etc/passwd with entries for just those two users, plus root. All other users (including the standard UNIX IDs: bin, sys, etc.) will be replaced.

Only use EXACT COPY if your profile contains all the users that need to be on a machine (see 8.2.9, “Deleting User Records” on page 286 for more information).

In the above scenario we were dealing with distribution to a Profile Manager, and we have seen how we can modify the user profile at each level. Distribution to a managed node (in our case, rs600019) involves a similar process. Distributing to rs600019 will send a copy of ITSO_Residents to rs600019 but will not update the system files.

Figure 261 on page 284 and Figure 262 on page 284 show the sequence of opening the subscriber icon for the managed node. You can see the copy of the profile displayed in the Managed Node window.

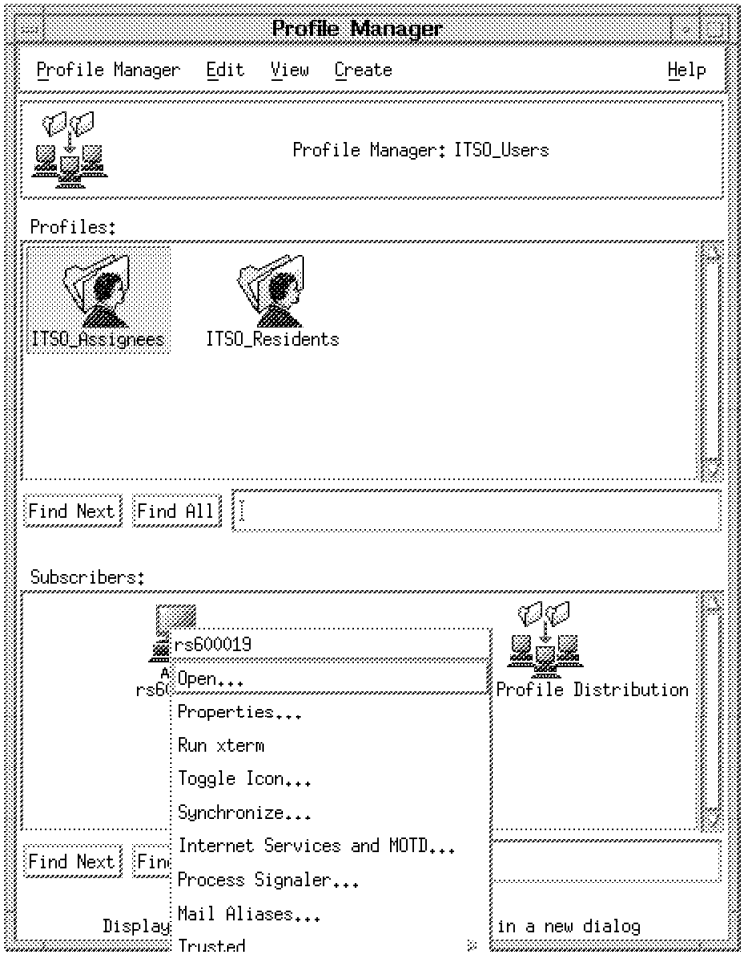


Figure 261. Open the Icon for Subscriber rs600019

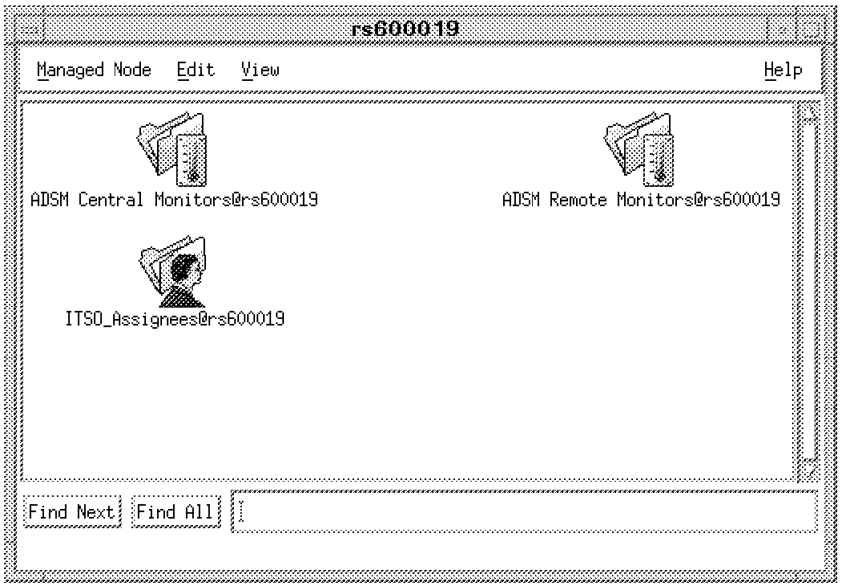


Figure 262. Subscriber's Copy of ITSO_Assignees

You can now make any required changes to the user record before selecting **Distribute** from the Profile menu. The result of doing so is shown in Figure 263

on page 285. We are now distributing the profile from the managed node to its own system files. As there are no further levels to go to, we are not prompted for **Next level of subscribers** or **All levels of subscribers**, only for **Preserve modifications** or **EXACT COPY**.

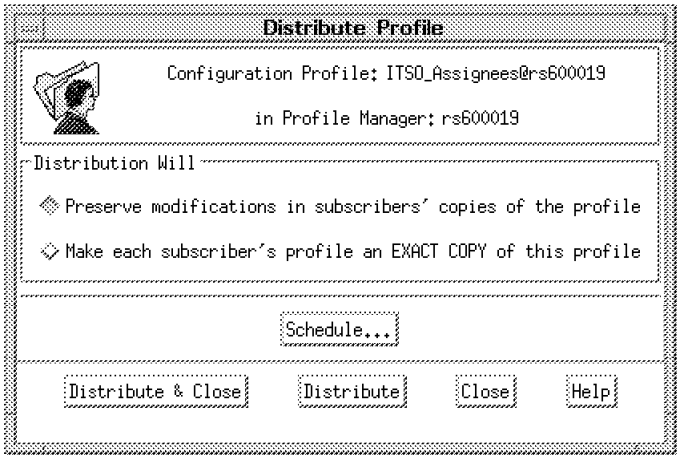


Figure 263. Distribution Options from a Managed Node

Once again, you will normally select **Preserve modifications** because EXACT COPY should only be used if the profile contains *all* of the users that need to be on a machine, not a subset as in this case. User *fred* will now be created on rs600019 with UID=500 and will include changes made to the record at the Managed Node level.

8.2.8 Modifying User Definitions

In the above example, we created a new user definition (fred) and distributed it to the end nodes. Modifying this user record is simply a matter of changing the definition at the appropriate level in the Profile Manager hierarchy and then redistributing it.

For example, you could perform the following steps:

1. Return to the ITSO_Users Profile Manager and open ITSO_Assignees User Profile.
2. Make some change to fred's record.
3. Select **Distribute** from the **Profile** menu and change the following values:
 - **Distribute To**: All levels of subscribers
 - **Distribution Will**: Preserve modifications

This distribution will update the system files on the subscribers (that is, the subscribers of the profile manager Test User Profile Distribution and the managed node rs600019). However, we have specified Preserve modifications, which means the change we made at the lower Profile Manager level to make UID=501 for fred will not be overwritten. rs600020 and rs600021 will remain UID=500.

One point to remember if changing a user's UID or GID, is to specify what to do with its files (see Figure 264 on page 286).

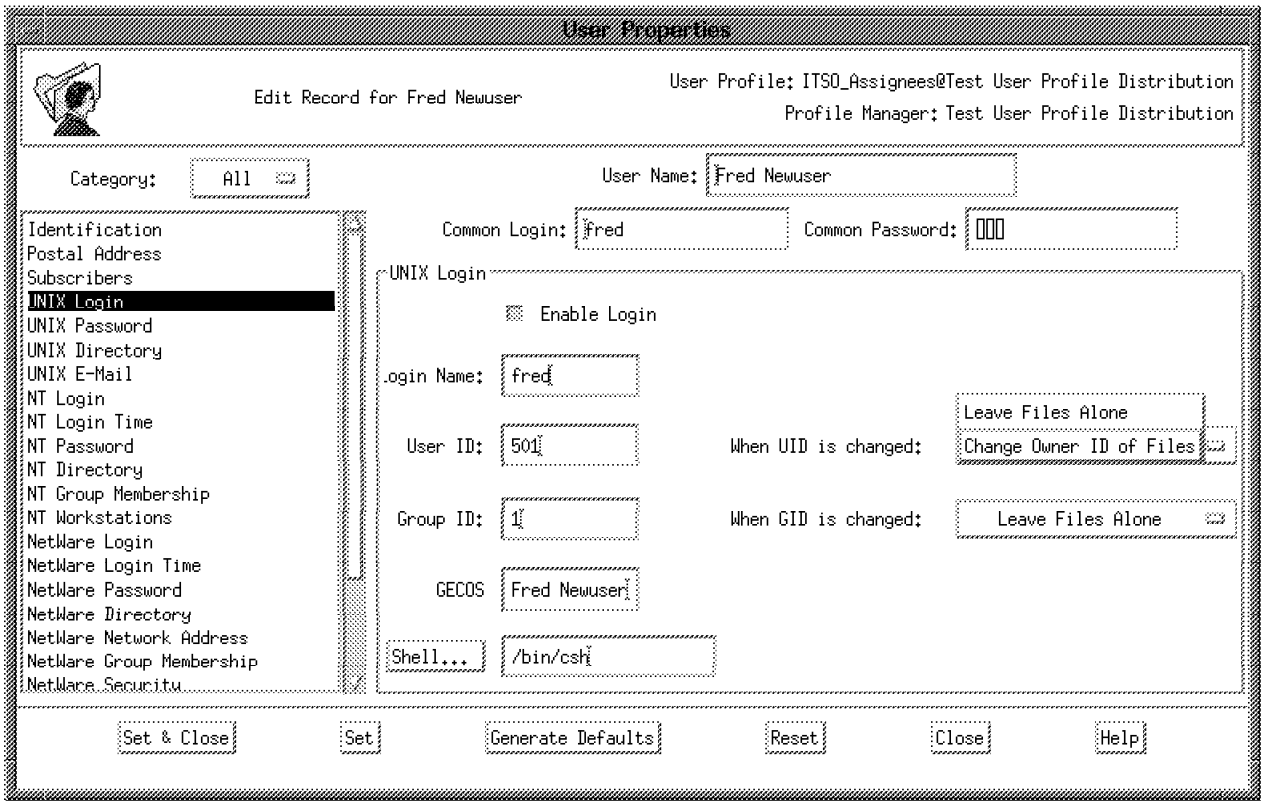


Figure 264. When UID Is Changed

8.2.9 Deleting User Records

We have shown how user records are created and modified. As you might expect, deletion of user records follows the same pattern of subscription and distribution. However, the detailed operation can be a little confusing so we have tried to explain it in some detail.

The *Tivoli/Admin User and Group Management Guide* says:

"If you no longer need a user account, you can delete it from the profile. When you distribute the profile again and use the option that makes subscribers' an exact copy of the profile, the record is deleted from all subscribers receiving the new copy. Otherwise the record is deleted only from the original profile and not subscribers."

Although this is true, it is not the whole story.

When a user profile is populated with the `wpopusrs` command or you add users manually to the profile, Tivoli/Admin is not yet really managing those users. Tivoli will start managing the records that you distribute with the EXACT COPY distribution option. Tivoli will also manage newly added or modified records in the profile that you distribute with distribution option Preserve Modifications.

- When you use the EXACT COPY option, the system files on the endpoints are completely replaced to match exactly what you have in the profile. This means that where you have a profile that contains a subset of users an exact copy will leave /etc/passwd with entries for just that subset, plus root. All other users will disappear. So be very cautious in using the exact copy

option. It will certainly delete users that you have removed from the profile, but it may remove more than you wanted! Only use EXACT COPY if your profile contains all the users that need to be on a machine.

- Users can also be deleted from the endpoint system files using the distribution option Preserve Modifications in certain circumstances.

 Deleting users from the profile before any distribution has been done will affect only the profile. If you delete a user from a profile that *has* been distributed at some time, the user will actually be deleted from the endpoint system files when you next distribute the profile. This applies even when using the preserve option.

 The result of this is that if you want to delete a user from the system files of managed nodes and the user record exists in a profile that contains only a subset of the users that exist on a machine, do the following:

 1. If this is a new user profile that has just been populated using the *wpopusrs* command or an existing profile that has had additional users appended, then make sure that Tivoli knows about the users, as follows:
 - Distribute the user profile to **Next level of subscriber** with **Preserve Modifications**.
 2. Delete the user records that you want to remove from the profile.
 3. Distribute the user profile to **All levels of subscriber** with **Preserve Modifications**.

 The users will be deleted from the system files.

8.2.10 How Tivoli Admin Updates System Files

When user records are distributed to the endpoints, (that is, the system files are updated) Tivoli/Admin creates a backup copy of /etc/passwd, /etc/security/passwd and /etc/security/user. These copies are put in the Tivoli database directory, in our case /var/spool/Tivoli/rs600024.db as:

```
/var/spool/Tivoli/rs600024.db/file_versions/etc/passwd
/var/spool/Tivoli/rs600024.db/file_versions/etc/security/passwd
/var/spool/Tivoli/rs600024.db/file_versions/etc/security/user
```

The updates to the files are handled using the Revision Control System (RCS). Tivoli provides commands for managing the operation of RCS:

- wco
- wci
- wident
- wrcs
- wrcsdiff
- wrcsmerge
- wrlog

These commands are documented in the *Tivoli Management Platform Reference Manual*.

8.2.11 Invoking Tivoli/Admin Functions from the Command Line

As in every Tivoli function, anything that can be done using the GUI desktop can also be done with a command. This makes it easy to incorporate functions into other processes, using shell scripts or other programs.

In this section we list a few of the available commands in the sequence that you would need them if you were to re-create the examples we have shown previously using the desktop.

Create a Profile Manager

```
wcrtprfmgr @PolicyRegion:rs600024-region ITSO_Users
```

Create User Profiles

```
wcrtprf @ProfileManager:ITSO_Users UserProfile ITSO_Assignees
wcrtprf @ProfileManager:ITSO_Users UserProfile ITSO_Residents
```

User Profile Policies

1. List attributes in a profile:

   ```
   wlspolm @UserProfile:ITSO_Assignees
   ```

 Some of the attributes you will see listed are uid, gid, city, subscribers and so on. This is equivalent to the information shown when you open a profile icon from the desktop.

2. Retrieve the current default policy for an attribute:

   ```
   wgetpolm -d @UserProfile:ITSO_Assignee uid
   ```

 This will display the script or constant associated with the attribute. To change a script, redirect the output of the script into a file:

   ```
   wgetpolm -d @UserProfile:ITSO_Assignee uid > /tmp/uid.attribute
   ```

 This command will return the *script arguments*:

   ```
   script arguments:  $real_name  $login_name
   ```

3. Change the default policy by editing the script you have retrieved and then replacing it, as follows:

   ```
   wputpolm -d @UserProfile:ITSO_Assignees uid < /tmp/uid.attribute
   ```

 If you want to change the script arguments:

   ```
   wputpolm -d -args='$real_name,$login_name' \
            @UserProfile:ITSO_Assignees uid  < /tmp/uid.attribute
   ```

 The arguments that you can use are those listed by the `wlsplom` command. If the new default policy is a Constant, enter a value:

   ```
   wputpolm -d -c TRUE @UserProfile:ITSO_Assignees passwd_aging
   ```

 Validation policies can be changed in the same way, but replace `-d` with `-v` in the commands.

Populate the Profiles: To populate the profile from existing user information:

```
wpopusrs -o -l @ManagedNode:rs600024 @UserProfile:ITSO_Assignees
```

This will populate the profile with all users on rs600024. To selectively populate the profile :

1. Create a file containing a list of users (one user per line) and then enter:

```
wpopusrs -f /tmp/listofusers -o -l  @ManagedNode:rs600024 \
        @UserProfile:ITSO_Assignees
```

The profile will contain only the users specified in the filename.

Setting Subscribers: To set subscribers at the Profile Manager level:

```
wsub  @ProfileManager:ITSO_Users @ManagedNode:rs600019 \
     @ProfileManager&:"Test User Profile Distribution"
```

To set subscribers at the User Record level:

```
wsetusrs -su rs600019,rs600020 @UserProfile:ITSO_Assignees rob
```

To check the list of subscribers:

```
wgetusr @UserProfile:ITSO_Assignees rob | grep Subscribers
```

Create a User: To create a new user, using all the defaults in the Default Policy:

```
wcrtusr @UserProfile:ITSO_Assignees "John Smith"
```

To specify a login value and subscribers:

```
wcrtusr -l john -su rs600019,rs600020  @UserProfile:ITSO_Assignees \
       "John Smith"
```

Distribute User Profiles: Setting distribution defaults can be done only from the TME desktop. To distribute a user profile:

```
wdistrib -l maintain -m @UserProfile:ITSO_Assignees \
        @ManagedNode:rs600019
```

This will distribute User Profile ITSO_Assignees to Managed Node rs600019, maintaining local modifications and distributing to all levels of subscribers.

To overwrite local modifications, insert:

```
-l over_all
```

The default is to maintain local modifications. To distribute to the next level of subscribers only, omit the `-m` flag.

To distribute all user profiles in a Profile Manager to another Profile Manager:

```
wdistrib -l maintain  @ProfileManager:ITSO_Users \
        @ProfileManager:"Test User Profile Distribution"
```

If no subscribers are specified, distribution will go to all subscribers.

Part 3. System Monitoring and Event Handling Applications

© Copyright IBM Corp. 1996

Chapter 9. TME 10 Distributed Monitoring

Much of the power of modern computer applications lies in the fact that processing is distributed among many server and client machines. Such applications can take advantage of dispersed processing power and the integration of multiple data sources.

From a management point of view, however, distributed applications lead to a number of headaches. One of these is the problem of monitoring. The systems manager wants to be ensured of the health of each component of the distributed system. This can put a strain on the support staff, who have the task of monitoring an increasing number of geographically dispersed systems.

TME 10 Distributed Monitoring provides facilities for monitoring many aspects of a managed system. This includes both system resources, for example CPU utilization, as well as application resources such as daemons and logfiles. It can also be configured to respond to certain system events, for example sending an e-mail message or forwarding an event to the T/EC server.

There are two parts to TME 10 Distributed Monitoring, the IBM Systems Monitor products and the Tivoli/Sentry product. In this chapter we will be dealing with the latter product, and we will refer to it as *Sentry* to avoid confusion.

Sentry is in two parts: an agent process called the *Sentry engine* which performs the monitoring functions on the target system, and a set of TME profiles and dialogs for defining the monitors. The distribution mechanism for the monitors uses, as you would expect, the standard TME management by subscription model. Figure 265 shows how a monitor is defined. The diagram shows the object request brokers on the server and client system, with their respective databases, and the Sentry monitoring engine on the client system.

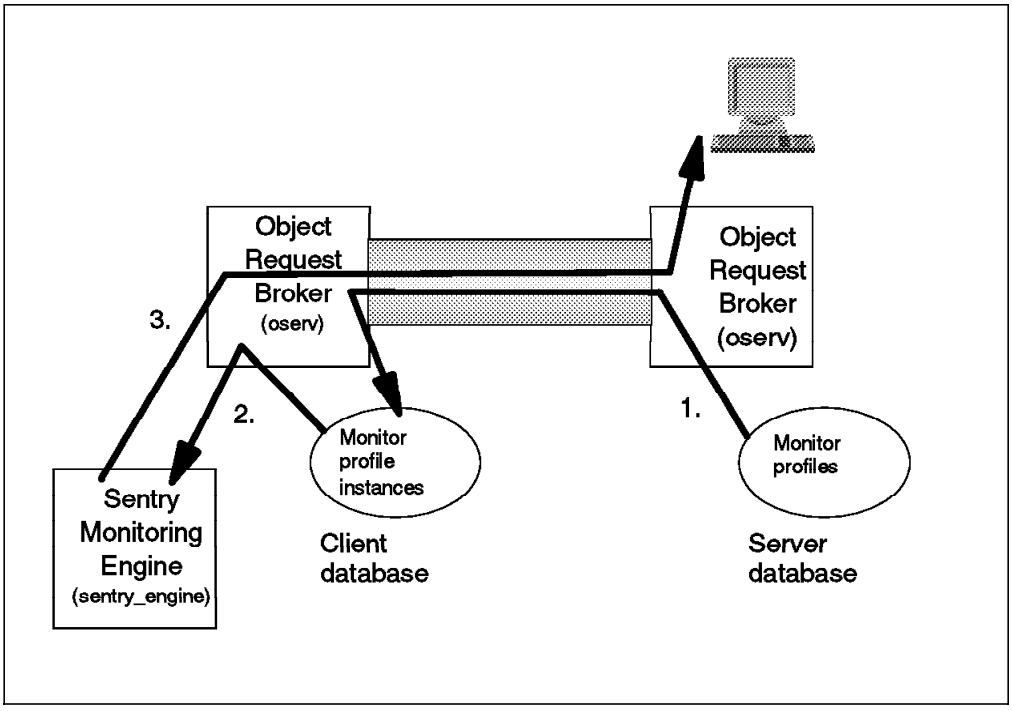


Figure 265. Operation of TME 10 Distributed Monitoring

© Copyright IBM Corp. 1996

The numbered arrows indicate the sequence of implementing a monitor:

1. The monitor profile is created and stored in the server database. At this point it is an operating system-independent description of the monitor to be executed. It is then distributed to the managed node using the normal TME management-by-subscription process.
2. The Sentry engine retrieves the profile from the client database and starts to perform the monitoring that it describes. The Sentry engine implements the monitoring collections for the specific operating system.
3. When the Sentry engine detects one of the conditions indicated in the monitor (for example, if a threshold is exceeded), it sends a request via the TME framework to trigger the specified notification actions. It may also execute some local automated action, depending on the response type.

For a full understanding of what Tivoli/Sentry can do, refer to the *Tivoli/Sentry User's Guide*.

9.1.1 Installing the Sentry Application

Sentry is comprised of a number of elements:

- The basic application, which provides the *Sentry monitoring engine* and the TME profile definitions
- A number of packages of monitors for different system environments, called *Monitoring Collections*.

Each of these elements is installed as a separate TME product, using the standard product installation process described in 5.2, "Installation" on page 109. Figure 266 shows the installation window, with the list of available Sentry components.

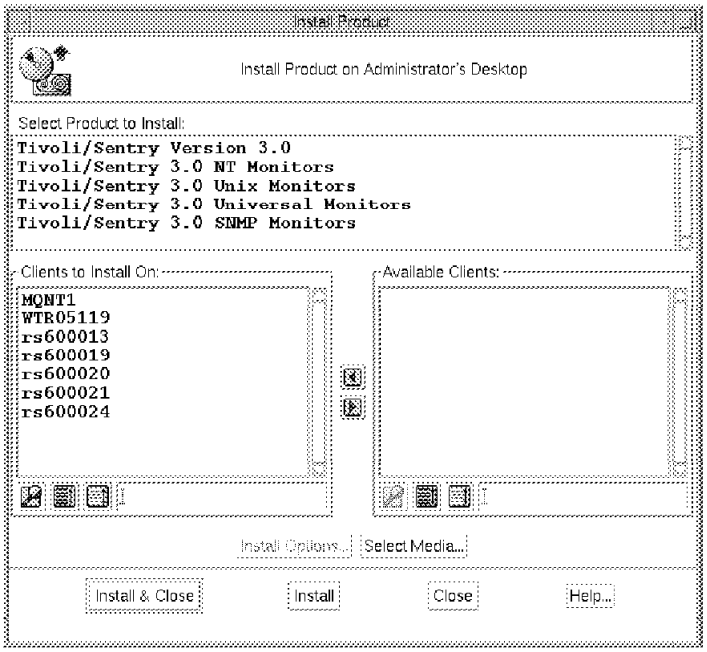


Figure 266. Components of TME 10 Distributed Monitoring

The more different monitoring packages you install, the more collections of monitors will appear in the Sentry profiles, as shown in Figure 267 on page 295.

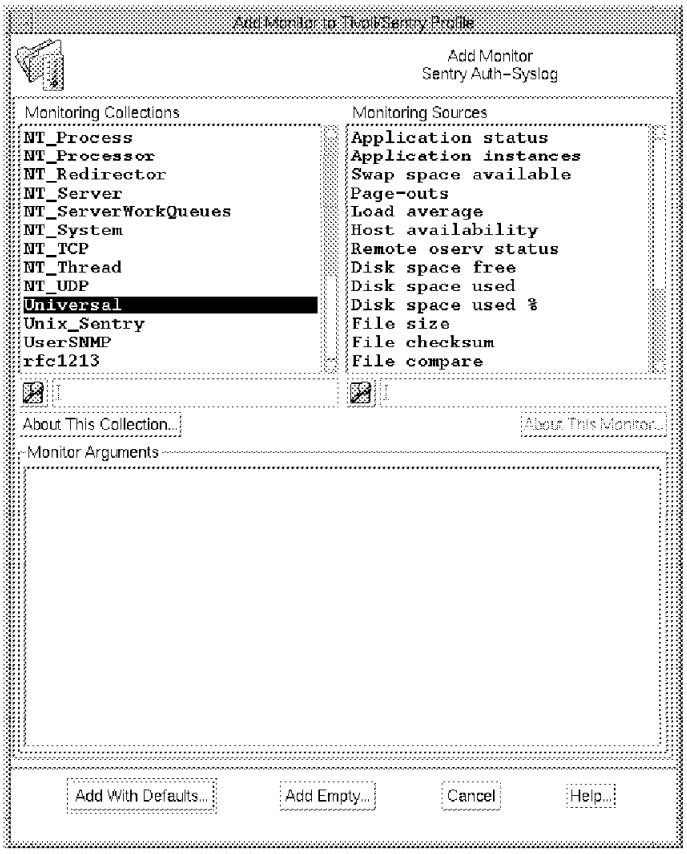


Figure 267. Viewing the Sentry Monitors

In this chapter we illustrate the Sentry capabilities by using them to monitor a specific application environment. The application chosen for monitoring is a World Wide Web server running on a RS/6000 machine.

9.2 Examples of Using the Sentry Monitors for a Web Server

We chose to monitor a Web server because it is a good example of an application that many customers are currently in the process of deploying, on a range of different operating systems and across distributed systems, many of which may not have a local administrator. The role that Sentry plays for us is to instrument the operation and performance of the Web server and the system it is running on.

9.2.1 Web Server Configuration

We used the IBM Internet Connection Web Server for our example. We will not discuss the installation of the server here, but we will mention a few important configuration changes that we had to make.

The Web server has one main configuration file, named /etc/httpd.conf. We made a number of modifications to this file. One modification was to change the values for the /private/* directory, to reset the user ID and password for administrator access from the system-supplied defaults.

The most important modification we had to make to /etc/httpd.conf was for the logfile parameters. These are defined in the standard password and group

definition file on the Web server. There are three logfile definitions and a report directory:

```
AccessLog        /usr/lpp/internet/server_root/logs/httpd-log
ErrorLog         /usr/lpp/internet/server_root/logs/httpd-errors
CacheAccessLog   /usr/lpp/internet/server_root/logs/cache-log

AccessReportRoot /usr/lpp/internet/server_root/pub/reports
```

There are a number of different roles that can be configured for a Web server; for example it could be a proxy Web server. However, in this case we ran it as a normal unsecured Web server without proxy functions or any other specialities. Our final package contained a number of different types of Sentry monitor. Several of them were based on monitoring messages written to the Web server log files, so as a preparatory step we had to route the log information to the syslog daemon. This is done by amending the LogToSyslog parameter in the http.conf configuration file, as follows:

```
LogToSyslog      On
```

This passes the messages to the syslogd daemon. In 9.2.6, “Monitoring Using the Asynchronous String Interface” on page 304 we show how to configure syslog to allow Sentry to pick up the messages.

9.2.2 Configuring the Sentry Monitors

Before configuring Sentry monitors, the Tivoli Management Platform must be installed on the TMR server and on the ManagedNode client where the monitoring is to take place. For our example the TMR server was rs600024 and the client running the Web server was rs600021. Both machines were RS/6000s. Next, the Sentry base code and the appropriate Monitoring Collections must be installed. The base code has to be installed on both the TMR server and the monitored node, but the monitoring collections are only installed on the TMR server. For the range of monitors that we wanted to use, we had to install two monitoring collections:

1. Tivoli/Sentry 3.0 UNIX Monitors
2. Tivoli/Sentry 3.0 Universal Monitors

If you select to install these products on a client node as well as the TMR server, you will not get an error message, but you will see messages that the installation will not have any effect on the target system.

Having installed all of the necessary Sentry code, the next stage is to create profile managers to contain the Sentry profiles.

9.2.3 Creating the Profile Manager

The first thing is to select the policy region in which you want to place the profile manager. Remember that the policy region sets the boundary of control for each TME administrator. Therefore the choice of policy region depends on how you plan to administer the environment:

- You could have one central operations organization that sets up all of the system monitors. In this case you would probably create one policy region hierarchy to which only that one group of administrators has the *admin* role.
- You could decide to have the setup of the monitors be treated as part of the application setup. In this case you would probably create a Web server

policy region, containing all of the profiles associated with implementing a Web server.

- You could have a distributed organization, in which case you would create policy regions that represented different parts of the company.

These are just a few of the possible configurations. If you are not sure which to apply, don't worry; you can always create the profile manager in one policy region and then drag it to another policy region if the administrative requirement changes.

To define the profile manager, do the following:

- Double click on the **Policy Region** icon.
- Select **Create** followed by **ProfileManager** (see Figure 268).

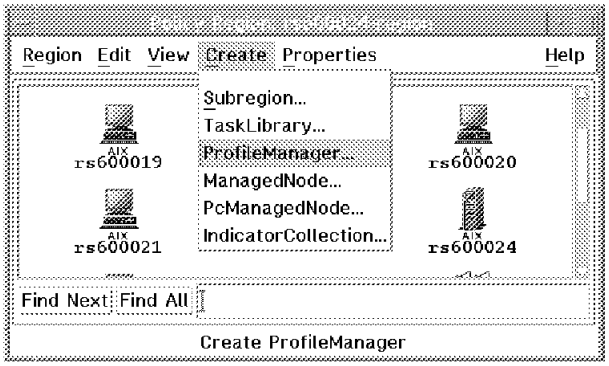


Figure 268. Create a Profile Manager

- Enter a name for the profile manager. We chose to call it *Internet-Prof-Mgr*.

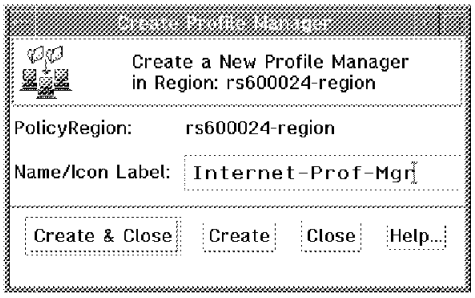


Figure 269. Create a Profile Manager. Put in the name/label.

- Select **Create & Close**.

The new icon will look like the one shown in Figure 270 on page 298.

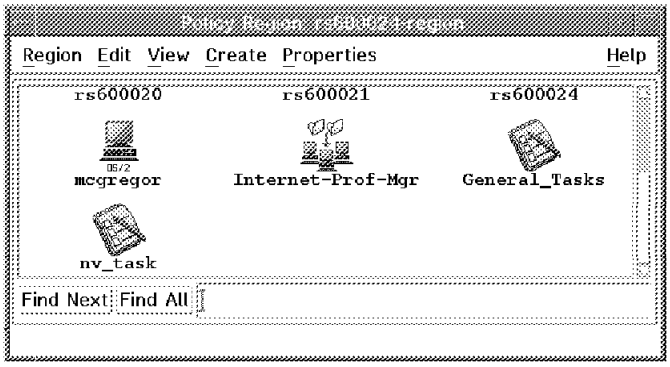


Figure 270. Policy Region with the New Profile Manager Icon

The next stage is to create the profiles.

9.2.4 Monitor Profiles

A profile manager can contain a number of profiles of different types. To create the profile we did the following.

- Double-click on the **Profile Manager** icon.
- Select **Create => Profile...**, as shown in Figure 271.

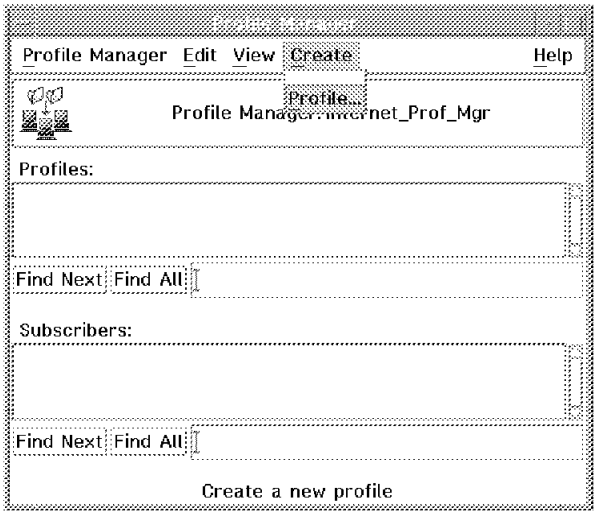


Figure 271. Create a Profile

- Double-click on **SentryProfile**.
- Add the name of the new profile. For our example we used *WWW-Profile*.

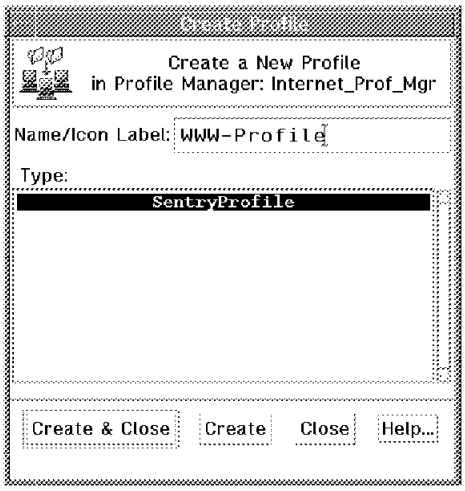


Figure 272. Create a SentryProfile

The new icon now appears in the Profile Manager window. You can view or edit the properties by using the context menu. At this moment, there is no data in the profile.

For this example we created three profiles:

Base In the base profile are monitors that could be used for any AIX system:

- Disk space free (/var).
- Disk space free (/usr).
- Host status (this checks the availability of a given node from the monitoring node. For example, you may want to check that the machine can talk to its default IP gateway).
- CPU load average.

WWW-Profile These monitors are a little more specific for the Web server:

- Status of the httpd (Web server) daemon.
- Disk space free in /usr/lpp/internet.

Auth-Syslog In this profile are monitors that are driven by asynchronous events:

- String script (filtercheck.ksh)
- Asynchronous string (root login)

Having created the profiles, the next step is to populate them with monitor definitions.

9.2.5 Monitors

The monitors are the fundamental part of the Sentry application. The monitors define what resource is to be monitored and what actions are taken if a threshold is exceeded or met. The normal actions involve alerting someone to the fact that a threshold has been triggered. However, automated responses can also be executed. Here is a summary of all possible actions:

Send Tivoli Notice This option posts a notice to a specified notification group, using TME services. You have to be subscribed to get the notice.

Popup Popup a window within a message. The default is that all administrators get this window, but you also can select some administrators. This action, too, uses TME services.

Change Icon Specifies whether the indicator icon is changed or not.

Task You can specify a task that can be released.

Send e-mail to An e-mail can be sent to a specific person, or to a group, by separating the addresses with commas.

Log to file Each pass of a threshold can be logged into a specific logfile.

Run program You can trigger a program or script that will be started up.

Send event to enterprise console Sentry provides an event adapter function for T/EC. See Chapter 11, “Tivoli/Enterprise Console Adapters” on page 355 for more details.

Apart from this you have to define ″Message Styles″, and set ″Distribution Actions″, ″Restrictions″ and ″Monitoring Schedules″. These parameters are important because of the amount of data that is to be stored and evaluated.

We give a detailed description of the setup process for just one of our monitors: the disk space free monitor. The other monitors are basically similar, although we discuss any unique features. In each case you have to open the profile and edit its properties.

Initially the profile properties will be blank (see Figure 273).

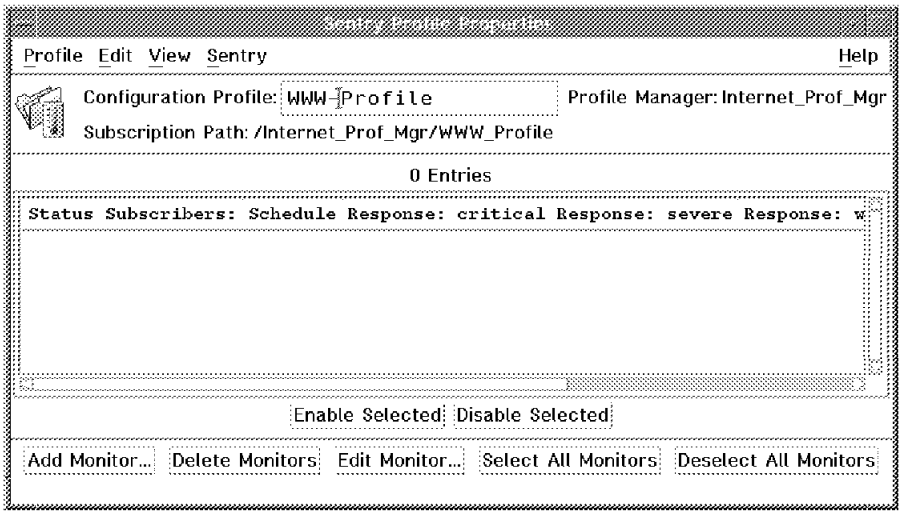


Figure 273. Edit the Properties of a Profile

9.2.5.1 Setting up a Disk Space Free Monitor

The disk space free monitor for the /var file system is an example of how to configure the disk monitors. The values must be adjusted for to each system and each file system. First of all you have to decide which action will be initiated on which threshold (a single monitor can contain several threshold settings). For the /var file system it might be something like this:

```
when space free < 1.0 MB result is ″critical″
when space free < 2.0 MB result is ″severe″
when space free < 4.0 MB result is ″warning″
otherwise result is ″normal″
```

In the editing panel of the monitor is another response level, which is called ″always″. You can use this to initiate actions at any time. We used this to track the actual severity in an indicator collection.

We decided to initiate the following actions on the response level ″critical″:

1. Send a Tivoli notice to Sentry
2. Pop-up a window for administrator's root and webadmin
3. Send an e-mail message to webadmin@rs600021.itso.ral.ibm.com
4. Log the threshold in a file (/tmp/www_monitor.log)
5. Extend the file system automatically by running the command `/etc/chfs/ -a size='+1' /var`
6. Send a critical event to the EventServer

We decided to monitor this file system every 30 minutes. In this case we did not restrict the time of day that the monitor would run, since the Web server should always be up and running.

We followed similar courses of action for the other threshold levels (severe and warning), but we reduced the severity of the alerts in each case, consistent with the severity of the event. For example, at the warning threshold we decided to send a pop-up window to the webadmin user and send a minor event to the T/EC server.

You can also choose the message style of the messages that are sent. There are three predefined styles:

1. Standard
2. Brief (one line)
3. Long - a very detailed description

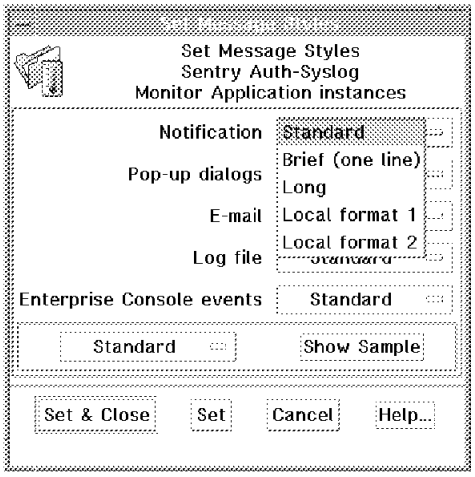


Figure 274. Set Message Styles

The disk space monitor is part of the Universal monitoring collection. You need to select this collection after clicking on the **Add Monitor** in the Profile Properties display, as shown in Figure 275 on page 302.

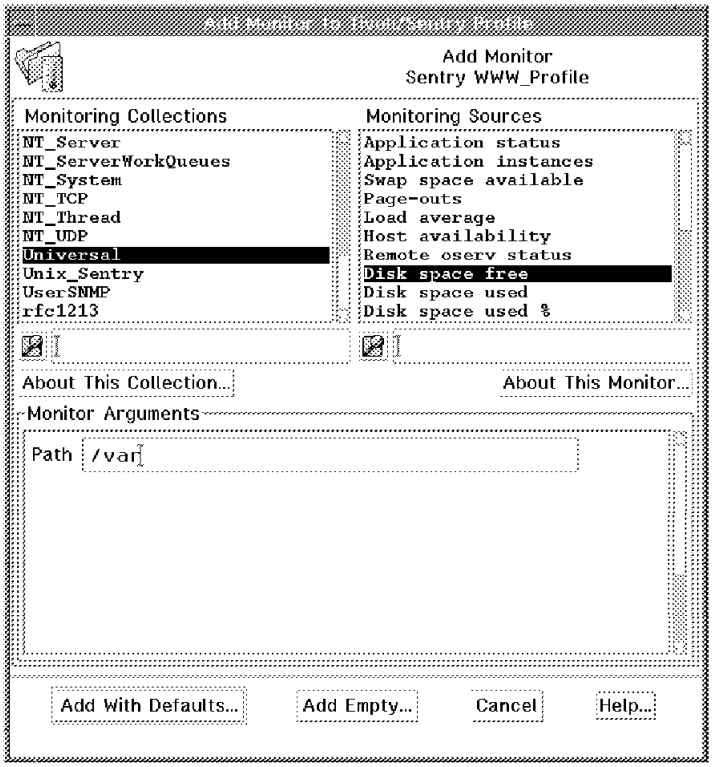


Figure 275. Select Monitor Type

Type in the file system you want to monitor and select **Add With Defaults**. Not every monitor has default values defined. In these cases, a dialog will prompt you to click on the **Add Empty** button.

The resulting panel is shown in Figure 276 on page 303. Note that each monitor can have several thresholds defined, but the dialog you see shows the details of only one of them at a time. In this case we are looking at the critical threshold definition. By clicking on the **Response Level** selector you can display the definitions for the other threshold levels.

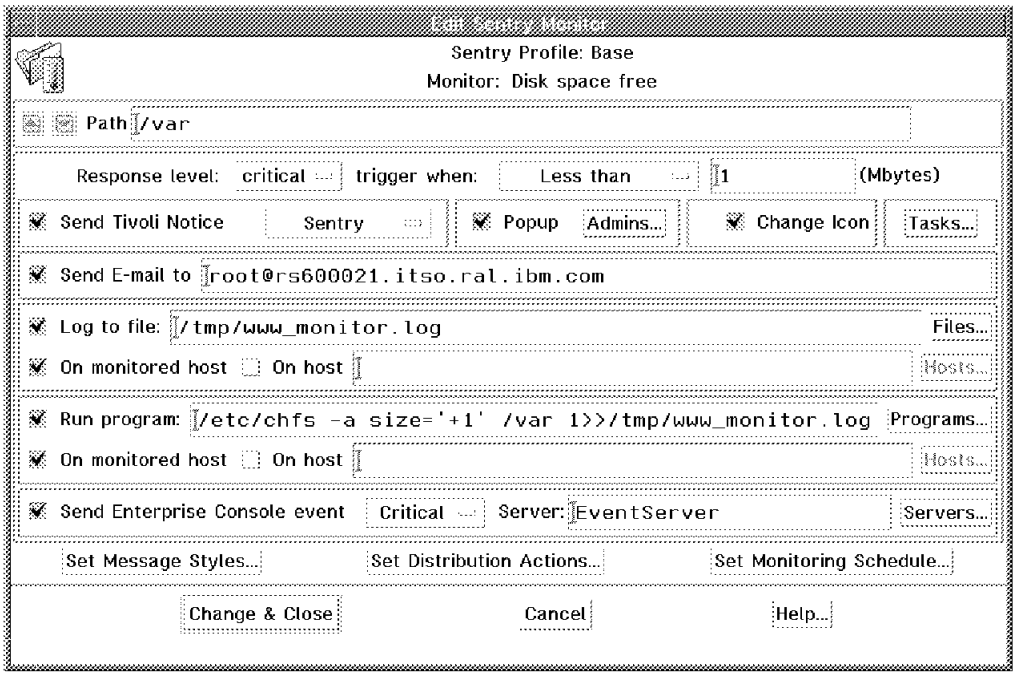


Figure 276. Edit Sentry Monitor

There are other properties that apply to the monitor as a whole, not just to specific threshold levels. These properties are defined by selecting the buttons at the bottom of the panel. The most important of them is the monitoring schedule. When you click on **Set Monitoring Schedule...** you will see the dialog shown in Figure 277, in which you define how often the monitor thresholds will be tested.

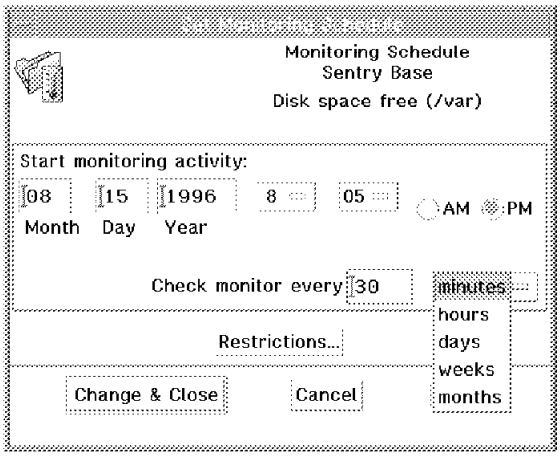


Figure 277. Set Monitoring Schedule

You can also define restrictions, for example you may only want to monitor during the day. Figure 278 on page 304 shows the available options.

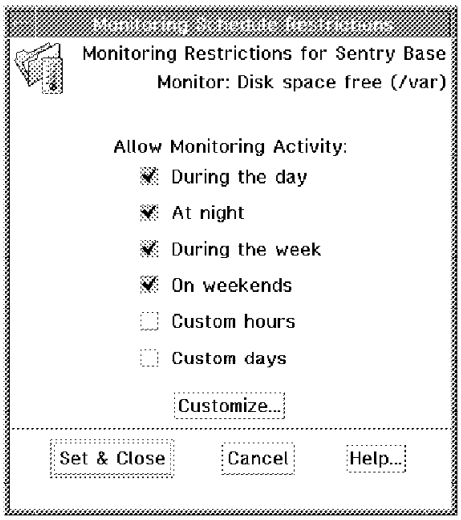


Figure 278. Restrictions

9.2.6 Monitoring Using the Asynchronous String Interface

Most of the monitors that we defined are similar in type to the disk space monitor described above. They are all based on a *polling model*: they check for a set of conditions at regular, timed intervals. Sentry also provides an asynchronous interface, in which the monitor is triggered by the arrival of a message, rather than by a timer expiring.

In fact, the asynchronous Sentry interface allows you to send any message string to the Sentry engine using the following line command:

```
wasync -c <channel> -s <data> -i <information> ·ManagedNode‟
```

The arguments in the command have the following meanings:

channel The message will be categorized with the channel argument, and will be used later when the Sentry profile is configured.

data The data can be a number or a string that will be used to make monitoring decisions.

information This argument can be anything at all. It is delivered through the engine in any e-mail or pop-up notices sent.

The asynchronous monitoring interface is more complex than the simple polling monitors and so there are some extra setup steps required:

1. Syslogd setup

 In our example we will get some information about root user ID logins, both direct logins and when users perform a `su root` command. This information is written to the syslog service automatically by AIX. We also want to get information from the Web server itself. We will get this from the user.info syslog category.

 First we have to edit the syslogd daemon configuration file /etc/syslog.conf. We added the following lines to the end of the file:

   ```
   # Send login and user information to a pipe for the Sentry engine:
   auth.info               /tmp/syslog.fifo
   user.info               /tmp/syslog.fifo
   EOF
   ```

The file identifier in these definitions can be a real file or a named pipe. When everything has been set up it saves disk space and maintenance headaches to use a pipe, but for testing we started out with a normal flat file. Before you restart the syslogd make sure that the pipe file has been defined. The command to create a named pipe is as follows:

```
mknod /tmp/syslog.fifo p
```

2. Creating a syslog filter file

 We need a script to filter the messages from the syslog daemon and select the items that we want to go to the Sentry engine. Figure 279 on page 306 shows the script. It uses pattern matching to search for the three messages that we are interested in and then executes the wasync command, passing an appropriate argument to Sentry.

```
#! /bin/ksh

#################################################################
#################################################################
##
##  Filename: syslog_filter.ksh
##
##  Date of the last modification: 08/12/96
##
##  Description: Korn-shell script to get information from the
##               syslogd via a pipe and put them into the
##               Sentry engine of the local machine
##
##
#################################################################
#################################################################



# defining some variables
. /etc/Tivoli/setup_env.sh
PIPE=/tmp/syslog.fifo


# get the information and filter them
exec < $PIPE
while read INPUT
do
  echo "ALL>>  " $INPUT
  case "$INPUT" in
    *su:\ from*to\ root*)
      echo "1>>  " $INPUT
      wasync -c "root login" \
             -s success \
             -i "$INPUT"
      logger ignore_me
      ;;
    *BAD\ SU*)
      echo "2>>  " $INPUT
      wasync -c "root login" \
             -s failure \
             -i "$INPUT"
      logger ignore_me
      ;;
    *NOT\ AUTHORIZED*)
      echo "3>>  " $INPUT
      wasync -c "web sniffer" \
             -s failedlogin \
             -i "$INPUT"
      logger ignore_me
      ;;
    *)
      ;;
  esac
done

exit
```

Figure 279. syslog_filter.ksh Script File

Having created the script, we started it in the background and restarted syslogd by using the `kill -1 <pid_of_syslogd>` command.

3. Setting up the asynchronous interface monitor

 Now we have got a script watching syslog and sending messages to Sentry. The last step is to set up Sentry so that our messages do not get lost. We added two monitors to a new profile using the Asynchronous String monitor from the Unix_Sentry monitoring collection. The monitor entries are

identified by the *channel name* passed by the wasync command. Figure 280 on page 307 shows the dialog to add a new channel definition.

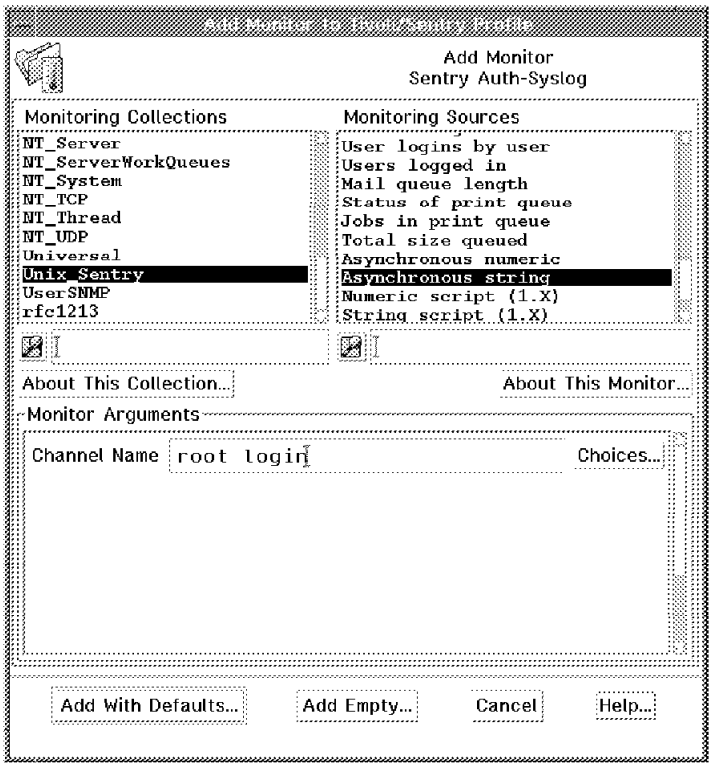


Figure 280. Add Monitor to Tivoli/Sentry Profile

The response level for the root login monitor is structured like this:

```
when =″success″ result is ″critical″
when =″failure″ result is ″severe″
otherwise result is ″normal″
```

For the Web sniffer monitor it is:

```
when =″failedlogin″ result is ″critical″
otherwise result is ″normal″
```

If you refer to Figure 279 on page 306 you will see that these values are what we passed in the data argument from the monitoring script.

Clearly, an asynchronous monitor of this kind is more complex to distribute and install than a normal polling monitor. For example, the monitoring script needs to be distributed and run, the named pipe must be created and the syslogd daemon has to be restarted. Sentry caters for this requirement by providing one or more *distribution actions* associated with the monitor. Figure 281 on page 308 shows the definition for one of these.

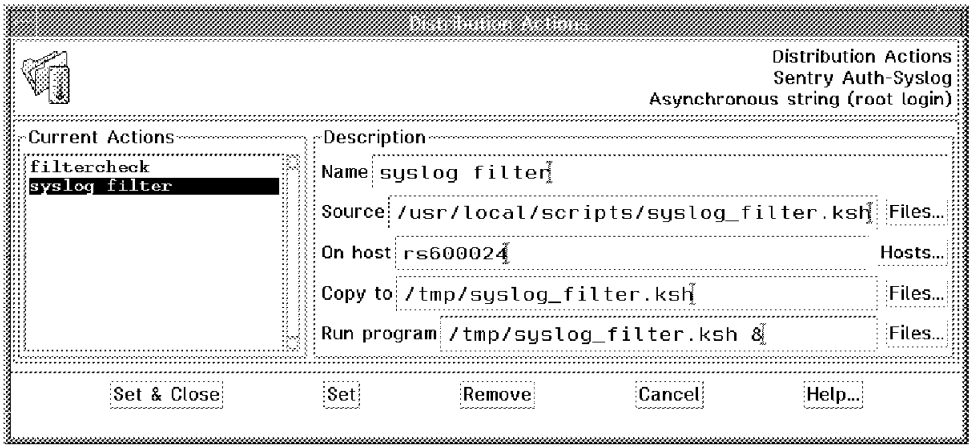


Figure 281. Set Distribution Actions Window

In total we created three such distribution actions:

a. The monitoring script itself, named syslog_filter.ksh

b. A script that kills any copies of syslog_filter.ksh that are running and then restarts it, named filterkill.ksh

c. A script that is invoked by a Sentry *String Script* monitor at regular intervals, named filtercheck.ksh

Table 11 summarizes the distribution actions.

Table 11. Distribution Actions on Profile

Profile name	Distribution name	Source	Copy to	Run program
Auth-Syslog	filtercheck	/usr/local/scripts/filtercheck.ksh	/tmp/filtercheck.ksh	
	filterkill	/usr/local/scripts/filterkill.ksh	/tmp/filterkill.ksh	/tmp/filterkill.ksh
	syslog filter	/usr/local/scripts/syslog_filter.ksh	/tmp/syslog_filter.ksh	

To summarize the distribution actions:

a. syslog_filter.ksh (the monitor script) is installed on the monitored node.

b. filterkill.ksh is also installed and executed when the monitor is first distributed. This creates the named pipe (if it does not exist), kills any running copies of syslog_filter.ksh, restarts it, and finally refreshes the syslogd daemon.

c. filtercheck.ksh is executed by another Sentry monitor at regular intervals to make sure that syslog_filter.ksh is running. The monitor is configured to run filterkill.ksh if it is not running.

There are other ways in which we could have achieved the same result. For example, we could have monitored the syslog_filter.ksh script using a Sentry application status monitor. Figure 282 on page 309 shows the filterkill.ksh script and Figure 283 on page 310 shows the filtercheck.ksh script.

```
#! /bin/ksh
#################################################################
#################################################################
##
##  Filename: filterkill.ksh
##
##  Date of the last modification: 08/16/96
##
##  Description: Korn-shell script to check the filterprogram
##               syslog_filter.ksh. It must not running twice or more
##               Though if it is so, all programs named syslog_filter.ksh
##               will be killed and one is restarted.
##
##  Arguments:
##
##  Authors: Bernd Kammholz
##
#################################################################
#################################################################

# defining some variables
PROGPATH=/tmp
PROGNAME=${PROGPATH}/syslog_filter.ksh

# Check that the named pipe exists
if · !-p /tmp/syslog.fifo “
then
  mknod /tmp/syslog.fifo p
fi

# grep for all PIDs of the program and kill them if there are more than
# one PID
PIDS=ps -ef | \
      grep $PROGNAME | \
      grep -v grep | \
      awk ’{print $2}’

for PID in $PIDS
do
  kill -9 $PID
done

$PROGNAME &

# refresh the syslogd
kill -1 cat /etc/syslog.pid

exit
```

Figure 282. filterkill.ksh File

```
#! /bin/ksh
#####################################################################
#####################################################################
##
##  Filename: filtercheck.ksh
##
##  Date of the last modification: 08/16/96
##
##  Description: Korn-shell script to check the filterprogram
##               syslog_filter.ksh. It must not running twice or more
##
#####################################################################
#####################################################################

# defining some variables
PROGNAME=syslog_filter.ksh
INSTANCES=ps -ef | \
      grep -v grep | \
      grep -c $PROGNAME

# put out the output of INSTANCES
echo $INSTANCES

exit
```

Figure 283. filtercheck.ksh File

9.2.7 Creating an Indicator Collection

One of the actions available in the Sentry monitors is to change the status of an *indicator icon*. These are special symbols that show the threshold status of a monitor using a thermometer metaphor. Indicator icons are placed in a special TME GUI collection known as an indicator collection. These are created within a policy region. As with any resource, before you can create an indicator collection you have to define that it is a managed resource type of the policy region, as shown in Figure 284 and Figure 285 on page 311.

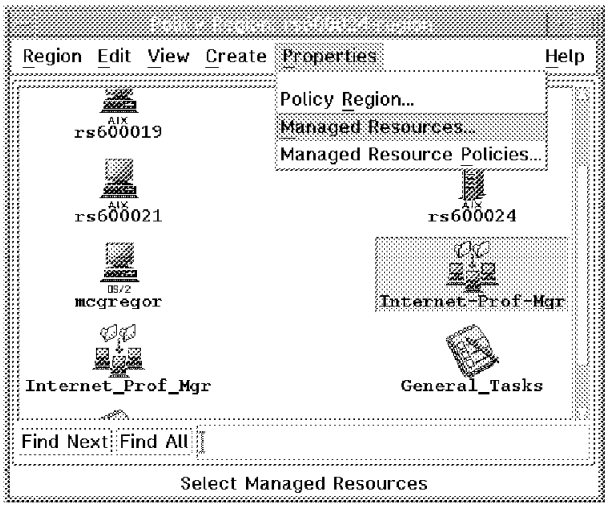


Figure 284. Defining Managed Resources for a Policy Region

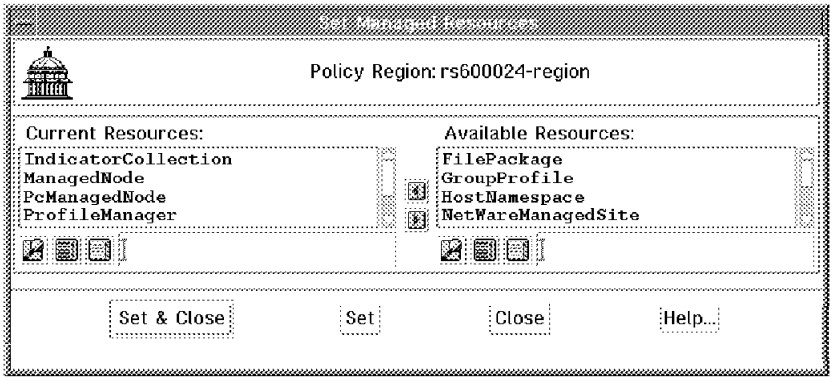


Figure 285. Set Managed Resources Window

You can then create a new indicator collection, as shown in Figure 286 and Figure 287.

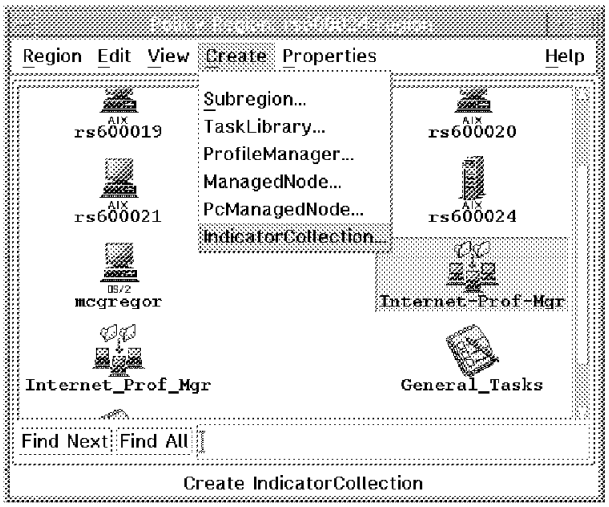


Figure 286. Create Indicator Collection

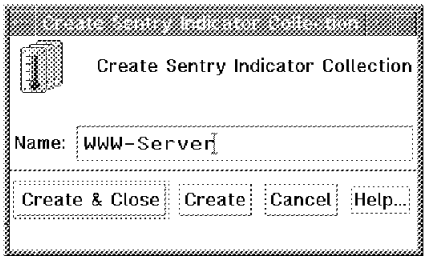


Figure 287. Define the Name of a New Sentry Indicator Collection

As the last step, you have to identify which indicator collection will represent each profile. In this case a sensible approach is to put all of the Web server monitors into one indicator collection (see Figure 288 on page 312 and Figure 289 on page 312).

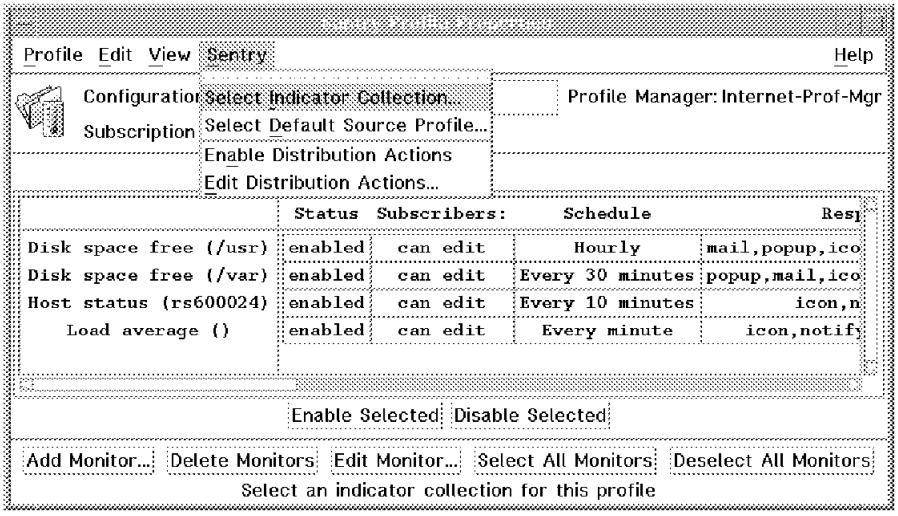


Figure 288. Select Indicator Collection

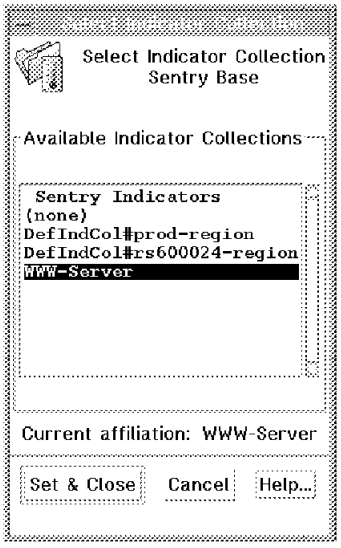


Figure 289. Select Web Server Indicator Collection

9.2.8 Preparing to Distribute the Profiles

Now we are ready to implement the monitors on our Web servers. To do this we make the Web server systems subscribers to the Sentry profile manager. In our case we only had one server, so we subscribed the managed node directly to the profile manager, but you may want to create a separate profile manager to which they are all subscribed. Then you would subscribe that profile manager to the Sentry profiles. This is the way that Net.Commander operates, as we can see in Chapter 14, “TME 10 Net.Commander” on page 441.

To add the subscribers, either drag the subscriber icons into the subscriber area of the profile manager, or select **Subscribers** from the profile manager menu shown in Figure 290 on page 313.

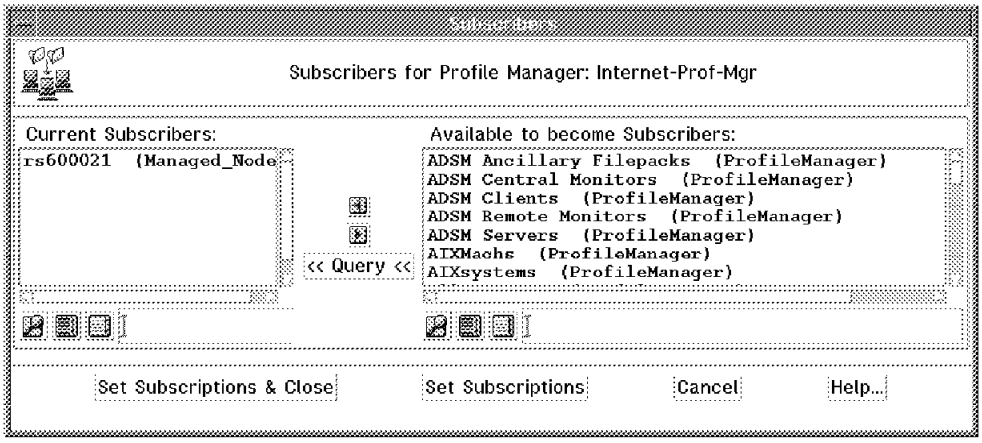


Figure 290. Select a Subscriber for the Profiles

9.2.9 Profile Distribution

Profile distribution for Sentry is the same as any other TME application. As we saw with User ID profiles in 8.2.7, “Distribute User Profiles” on page 276, the distribution process allows you to specify whether the profiles may be overridden at a lower subscription level, and whether the target nodes should be an exact copy of the profile, or whether modifications should be preserved. It is much easier to make this decision for Sentry than it was for Admin. In general you want to accept the default settings, which will make the target system profiles an exact copy each time you distribute.

Select one or more subscribers and profiles and open the distribution window.

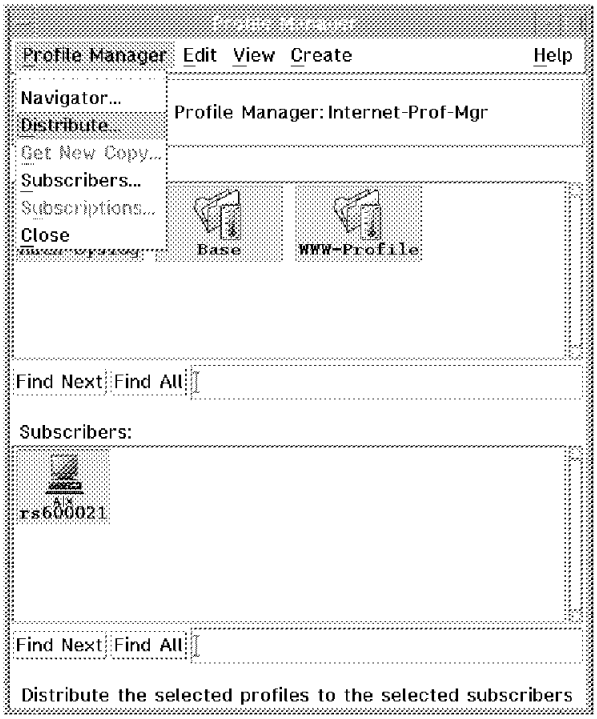


Figure 291. Distribute the Profiles

You will get a window on which you can decide whether to distribute now or schedule the distribution (see Figure 292 on page 314).

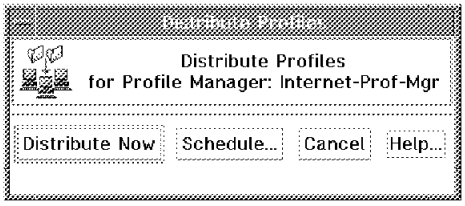


Figure 292. Distribute the Profiles

9.3 Planning the Sentry Monitors

Even our relatively simple requirement for monitoring the Web server turned out to involve a number of separate Sentry monitors. With time this number would probably grow, as we recognized additional aspects of the server that we wanted to keep an eye on. To keep track of the monitoring environment we recommend maintaining a table of the monitors that you implement and the actions defined for them. The process of creating the table also acts as a good stimulus for thinking about what kind of problems you should monitor for.

Table 12 on page 315 shows the monitoring plan for our Web server environment. You can see the details of the disk space free, daemons, logins, failed logins and load average monitors. Each monitor is subdivided into response levels. These are critical, severe, warning, normal and always. For each response level you can define different actions.

Table 12 (Page 1 of 3). Sentry Monitor Planning Table

What to monitor	Response level	Trigger when?	Send Tivoli notice	Pop-up on which Admins?	Change Icon?	Task	Send e-mail to	Log to file Where? On monitored host?	Run program on monitored host	Send enterprise console event On which console?	Message Styles	Set Dist. Actions	Monitoring Schedule	Restrictions
daemon: httpd	Critical	becomes unavailable	Sentry	root webadmin			webadmin@\ rs600021.\ itso.\ ral.ibm.com	/tmp/www_\ monitor.log	/usr/bin/\ startsrc -s httpd 1>>\ /tmp/\ www_\ monitor.log 2>>\ /tmp/www_\ monitor.log	Critical on EventServer	default	standard	3 min	always
	Severe	Down or unavailable	Sentry	webadmin				/tmp/www_\ monitor.log		Warning on EventServer				
	Warning	becomes available	Sentry	root webadmin				/tmp/www_\ monitor.log		Harmless on EventServer				
	Normal													
	Always				Yes									
disk space free: /usr/lpp/\ internet	Critical	< 2MB	Sentry	root webadmin			webadmin@\ rs600021.\ itso.\ ral.ibm.com	/tmp/www_\ monitor.log	/etc/chfs -a size='+1' /usr/lpp/\ internet	Critical on EventServer	default	standard	30 min	always
	Severe	< 5MB	Sentry	root webadmin				/tmp/www_\ monitor.log		Minor on EventServer				
	Warning	< 10MB		webadmin						Warning on EventServer				
	Normal													
	Always				Yes									
disk space free: /var	Critical	< 1MB	Sentry	root webadmin			webadmin@\ rs600021.\ itso.\ ral.ibm.com	/tmp/www_\ monitor.log	/etc/chfs -a size='+1' /var	Critical on EventServer	default	standard	30 min	always
	Severe	< 2MB	Sentry	root webadmin						Minor on EventServer				
	Warning	< 4MB		webadmin						Warning on EventServer				
	Normal													
	Always				Yes									

Table 12 (Page 2 of 3). Sentry Monitor Planning Table

What to monitor	Response level	Trigger when?	Send Tivoli notice	Pop-up on which Admins?	Change Icon?	Task	Send e-mail to	Log to file Where? On monitored host?	Run program on monitored host	Send enterprise console event On which console?	Message Styles	Set Dist. Actions	Monitoring Schedule	Restrictions
disk space free: /usr	Critical	< 5MB	Sentry	root webadmin			webadmin@\ rs600021.\ itso.\ ral.ibm.com	/tmp/www_\ monitor.log	/etc/chfs -a size='+1' /usr	Critical on EventServer	default	standard	60 min	always
	Severe	< 10MB	Sentry	root webadmin				/tmp/www_\ monitor.log		Minor on EventServer				
	Warning	< 20MB		webadmin						Warning on EventServer				
	Normal													
	Always				Yes									
Host status: rs600024	Critical	becomes unavailable	Sentry	root webadmin			webadmin@\ rs600021.\ itso.\ ral.ibm.com	/tmp/www_\ monitor.log		Critical on EventServer	default	standard	10 min	always
	Severe													
	Warning	becomes available	Sentry	root webadmin				/tmp/www_\ monitor.log		Harmless on EventServer				
	Normal													
	Always				Yes									
Load average	Critical	> 0.7	Sentry	root webadmin			webadmin@\ rs600021.\ itso.\ ral.ibm.com	/tmp/www_\ monitor.log		Critical on EventServer	default	standard	1 min	always
	Severe	> 0.6	Sentry	root webadmin				/tmp/www_\ monitor.log		Minor on EventServer				
	Warning	> 0.5		webadmin						Warning on EventServer				
	Normal													
	Always				Yes									
Asynchronous string: root login	Critical	=″success″	Sentry	webadmin			webadmin@\ rs600021.\ itso.\ ral.ibm.com	/tmp/www_\ monitor.log		Warning on EventServer	default	see distribution actions		
	Severe	=″failure″	Sentry	webadmin			webadmin@\ rs600021.\ itso.\ ral.ibm.com	/tmp/www_\ monitor.log		Critical on EventServer				
	Warning													
	Normal													
	Always				Yes									

Table 12 (Page 3 of 3). Sentry Monitor Planning Table

What to monitor	Response level	Trigger when?	Send Tivoli notice	Pop-up on which Admins?	Change Icon?	Task	Send e-mail to	Log to file Where? On monitored host?	Run program on monitored host	Send enterprise console event On which console?	Message Styles	Set Dist. Actions	Monitoring Schedule	Restrictions
Asynchronous string: web sniffer	Critical	= "failedlogin"	Sentry	webadmin			webadmin@\ rs600021.\ itso.\ ral.ibm.com	/tmp/www_\ monitor.log		Fatal on EventServer	default	see distribution actions		
	Severe													
	Warning													
	Normal													
	Always				Yes									
String script: /tmp/\ filtercheck\ .ksh	Critical										default	standard	1 min	always
	Severe	> 1	Sentry	webadmin			webadmin@\ rs600021.\ itso.\ ral.ibm.com	/tmp/www_\ monitor.log	see distribution actions					
	Warning	= 0	Sentry					/tmp/www_\ monitor.log	/tmp/syslog_filter.ksh					
	Normal													
	Always				Yes									

Chapter 10. Introduction to the TME 10 Enterprise Console

The TME 10 Enterprise Console (T/EC) provides a centralized operational environment for managing alerts generated from a large number of devices. Having one central point for all of the application, device and network alerts allows more flexible planning and control of help desk and problem management functions.

The T/EC is comprised of three main components:

Event Server The event server manages all the events generated within the network. The server uses a relational database to store the event information. There are several separate processes within the server responsible for different aspects of the function: event reception, rules processing and communication with the consoles, and task execution.

Event Console The event console is the graphical interface that displays the event information to the user. Multiple users can have an event console active at one time and the consoles can be distributed among a number of systems. The interface also allows the user to perform actions on individual events or groups of events. Any action taken by one user is reflected in the console of all the other active users.

Event Adapters The event adapters are installed on the remote managed machines and provide the mechanism to generate events and forward them to the event server.

Figure 293 on page 320 illustrates these components and the relationships between them.

© Copyright IBM Corp. 1996

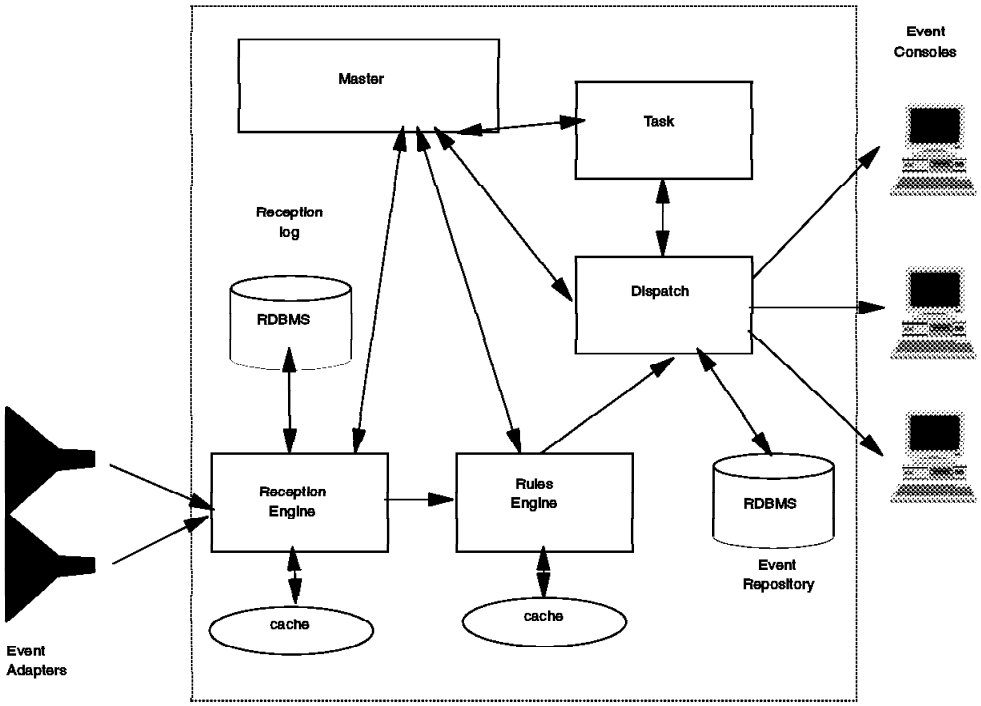


Figure 293. Components of the Tivoli Enterprise Console

The T/EC design provides a number of built-in features, for example:

- Ability to cope with a potentially high number of alerts
- Correlation between events that are separated in time and from different sources
- Automation of tasks
- Customized consoles for different operational requirements
- Filtering of unwanted events
- Alert escalation to e-mail, pagers and problem management applications

The T/EC server can receive events from a number of sources, these are covered in Chapter 11, “Tivoli/Enterprise Console Adapters” on page 355. The T/EC server also provides a generic link for problem management systems. We show an example of one such system in 13.1.1, “The Remedy Action Request System” on page 428.

10.1 Planning the Installation

We recommend an initial planning phase before installing the T/EC server application. This involves deciding what event sources you are interested in, which machine is to be the server and where it will be located. A potentially large amount of incoming traffic will be generated to the server machine.

For the purposes of this project, we decided to load the T/EC software as follows:

- rs600020 as the T/EC Server

- rs600021 as a T/EC Console
- rs600019 as a T/EC console connected to the server on rs600020

We also recommend that the T/EC server run on a dedicated machine, to allow the best flexibility in handling the erratic workload that it inevitably generates.

The next issue is what size database to create for the T/EC events. In fact, it is not critical that you get the correct size in the beginning because the size of the database can be extended. However, if the T/EC database becomes full the T/EC may behave in a confusing way, so it is better to over-allocate if you have the disk space available.

10.2 T/EC Installation

The full description of the T/EC server installation can be found in the *Tivoli/Enterprise Console User's Guide Volume 1*. Essentially it is like any other TME product installation, as described in 2.5, "Product Installation" on page 40. To install the event server you have to install two components, first the RDBMS Support application and then the Enterprise Console itself. The RDBMS Support component is, in fact, a run-time version of the Sybase relational database manager in the current version. Other database options will be available in future releases of T/EC. Figure 294 shows these two components as they appear when you select the T/EC installation CD from the product install dialog.

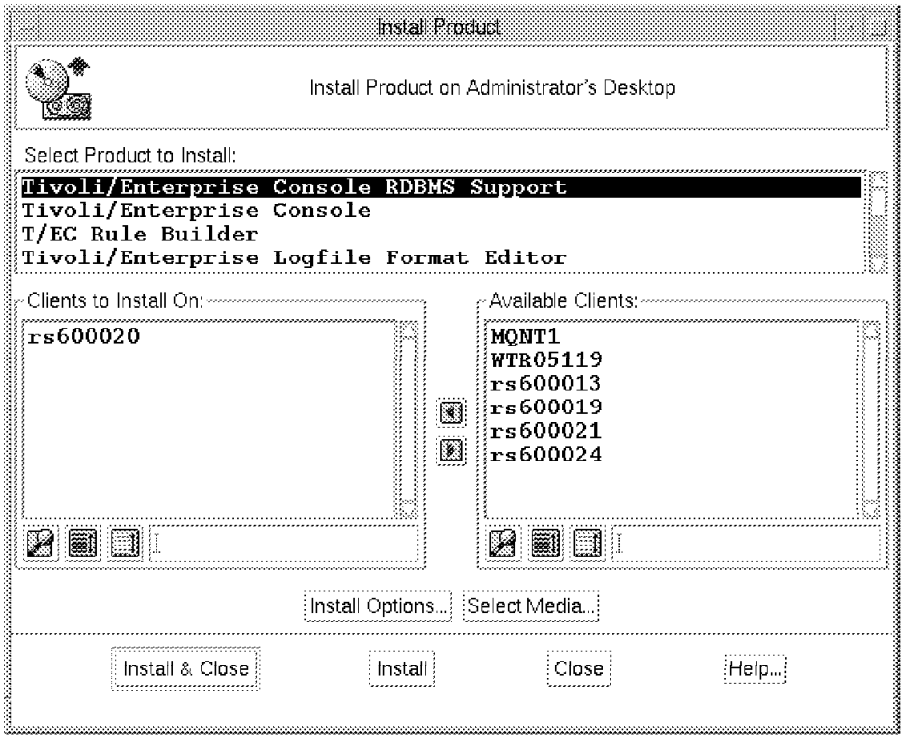


Figure 294. Installing the T/EC Components

When you select the RDBMS Support component you will be prompted for database configuration options, as shown in Figure 295 on page 322. You can also alter these values by selecting **Install Options** from the main product install dialog.

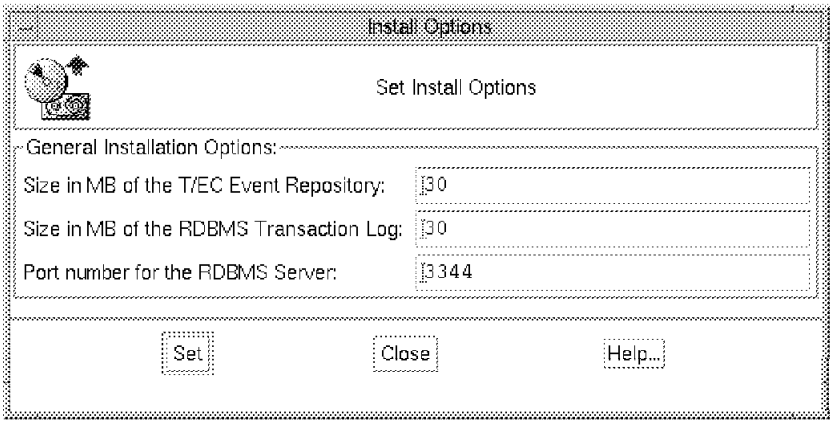


Figure 295. Configuring Database Options

Having installed the RDBMS component on rs600020, as shown above, we then installed the Tivoli/Enterprise Console component, also on rs600020. Finally we installed the Tivoli/Enterprise Console component alone on machine rs600021. No additional software was installed on rs600019.

While installing the T/EC software on rs600021, the message shown in Figure 296 on page 323 was displayed. This messages is warning you that a T/EC server is already installed for the T/EC region. The T/EC architecture only permits one server to be active within a TMR. In this case, however, we only wanted to be able to run the console software (that is, the GUI) on rs600021, so we can safely ignore this message.

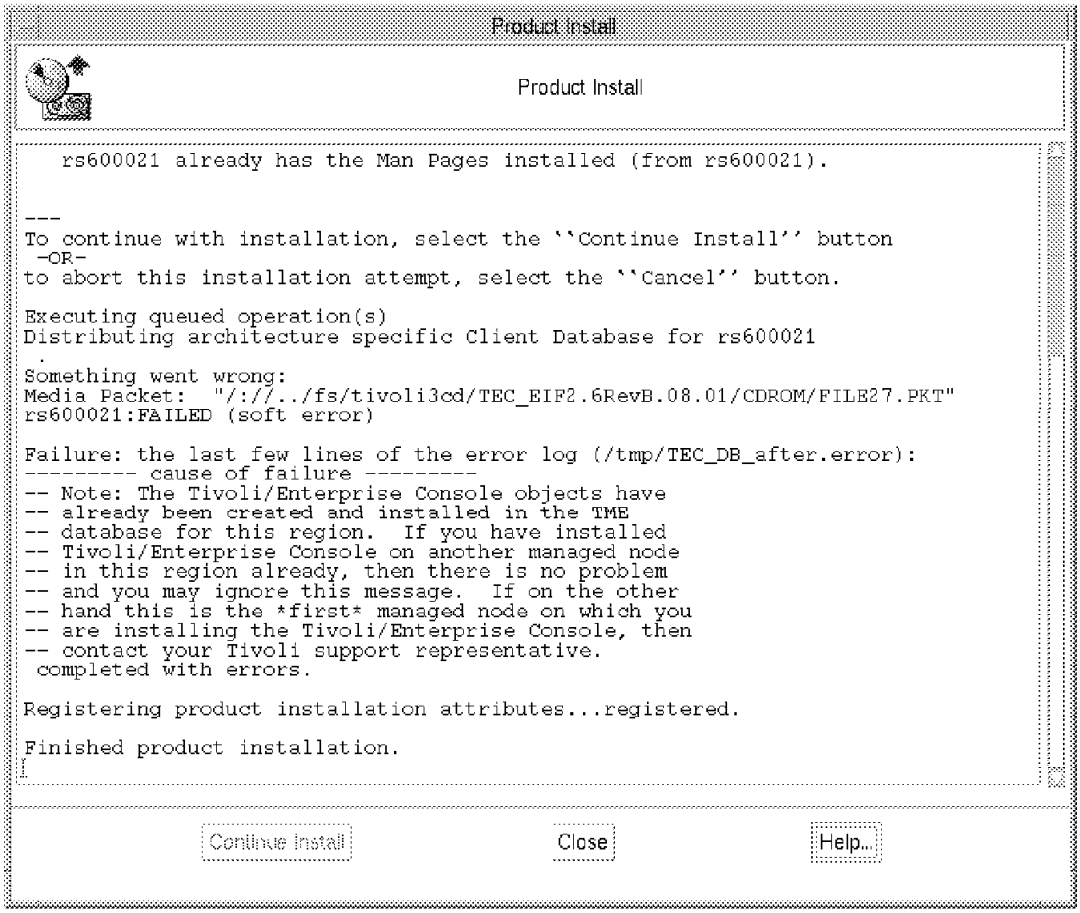


Figure 296. T/EC Installation Message

Once the installation is complete, the Event Server icon appears on the root Tivoli desktop.

10.3 Configuring the T/EC

There are four configuration steps necessary before the T/EC can be used:

1. Defining T/EC administrators
2. Load the rulebase
3. Define Event Sources and Groups
4. Configure the administrator consoles

There are other aspects of T/EC configuration, such as installing event adapters, creating event rules and setting T/EC server parameters. We will deal with these topics in later sections.

The easiest way to understand what you are trying to achieve when configuring the T/EC server, is to think of it as being in three parts, *preparation*, *input* and *output*. Preparation is simply defining the administrators that will use and configure the T/EC. Configuring the input involves defining which event types the T/EC server can expect to receive, and how it should interpret them. When configuring the output you are defining which administrators can run the T/EC console, and what subset of the total events they will see.

10.3.1 Defining the Administrators

In our case, we created an environment that allowed three different levels of operators to use the T/EC. Two of the administrators have the ability to modify the behavior of the T/EC, and one is purely a console user. We also define different event views for the users, by allocating different groups of messages to each. Note that for each user ID you can have multiple copies of the TME desktop active at one time, but only *one* copy of the event console can be running.

The consoles we created are for the administrators shown below:

- Root_rs600024-region on rs600020
- Paul on rs600021
- SNMP_MANAGER on rs600019

First we create the TME roles to allow an administrator to be able to execute and configure the T/EC console. The process for defining administrators is described in 3.1.7, “Create Administrators” on page 70.

To configure the Administrators, we perfomed the following tasks from the Tivoli desktop.

- Double-click on the **Administrators** pull-down menu.
- Then click on the administrator icon (in our case, **Root_rs600024-region**), and select **Edit Resource Roles**, as shown in Figure 297.

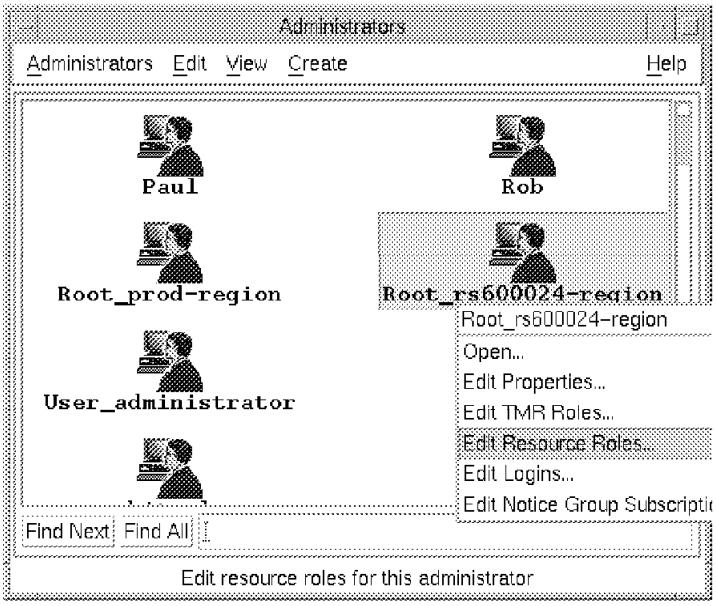


Figure 297. Configuring an Administrator

- Define the appropriate current roles for the *EventServer* resource, as shown in Figure 298 on page 325.

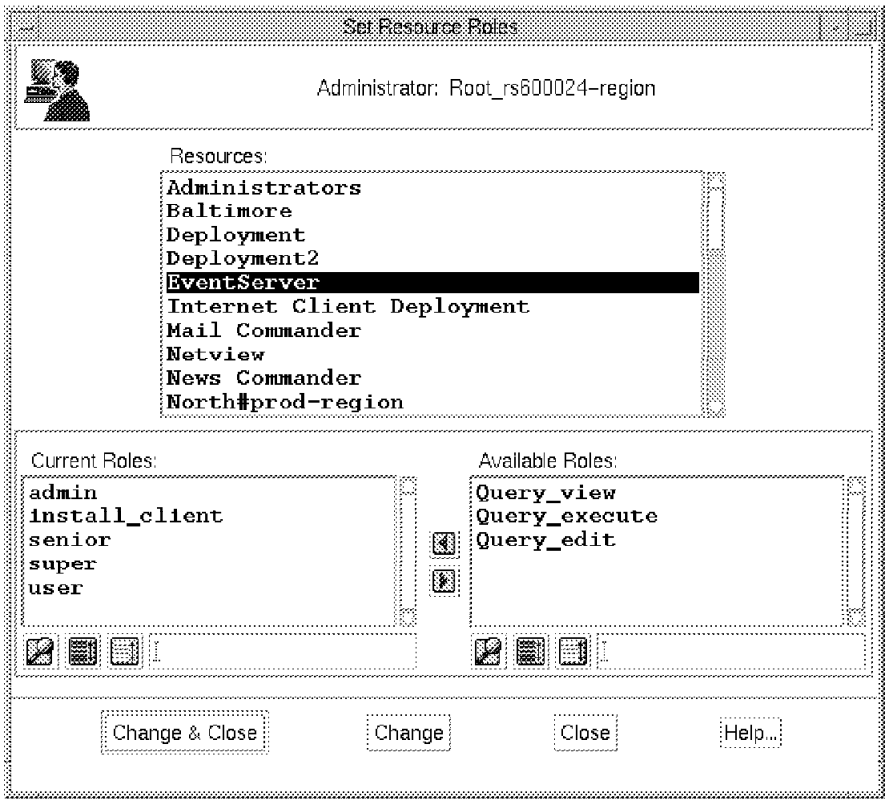


Figure 298. Selecting Administrator Roles

- Select **Change & Close**.

The current roles to select depend on what capabilities you want the administrator to have. We wanted administrator SNMP_MANAGER to be able to view, acknowledge and close events, but not able to control the operation of the event server or reconfigure it. Therefore, we selected only the user and admin roles. For admistrator Paul we defined all roles, giving him full control. Refer to the *Tivoli Event Console Users Guide* for the roles required for each operation.

Once the administrators have permission to use the T/EC, we can create consoles for them. We performed the following process for each of the administrators:

- Double-click on the **Administrators** pull-down menu and locate the administrator (see Figure 299 on page 326).

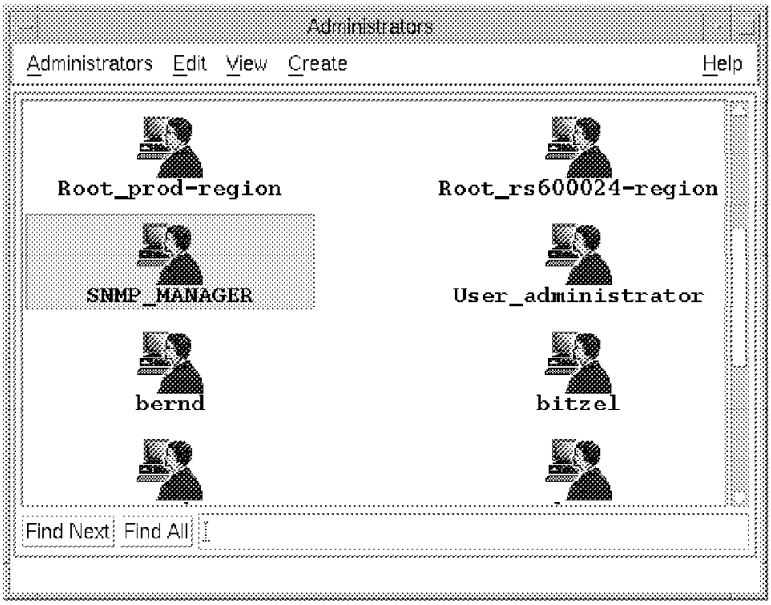


Figure 299. Creating the Event Console for SNMP_MANAGER

- Double-click on the icon to open the administrator's desktop and then select **Create** followed by **Event Console** from the menu bar (see Figure 300 on page 327). If you do not see the Event Console option after installing T/EC, you may have to shutdown all TME desktops on the managed node and restart them. The result of this is to recycle the uiserver process, which manages GUI interactions.

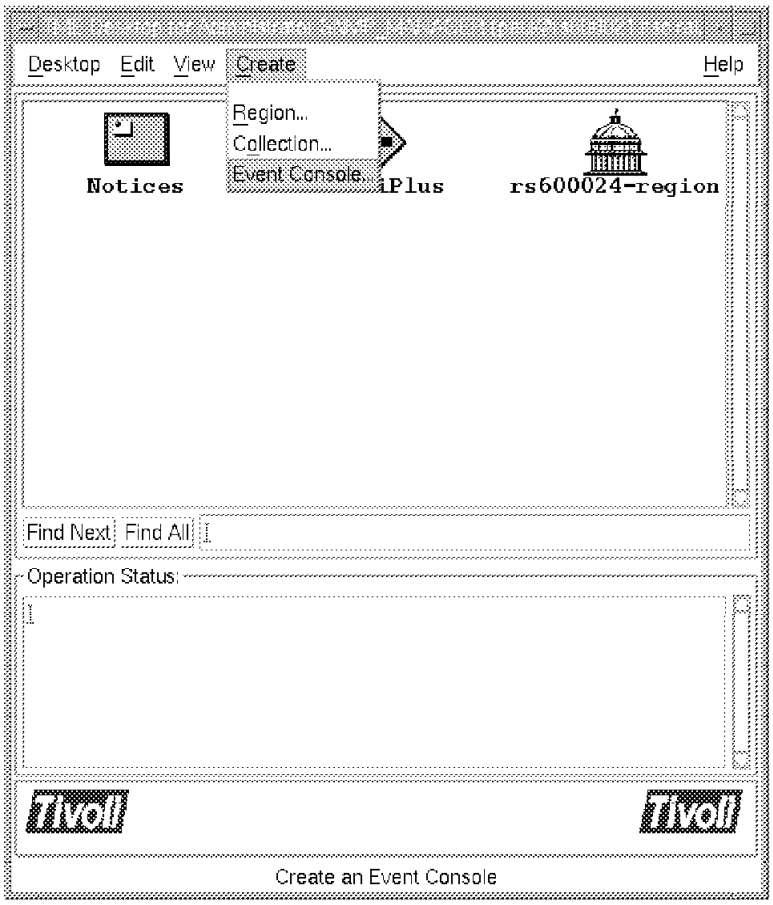


Figure 300. Creating the Console for SNMP_MANAGER

- Select the host where the console will be executed (Figure 301). This can be either the event server itself or a managed node where you have installed the T/EC software. Click on **Create**.

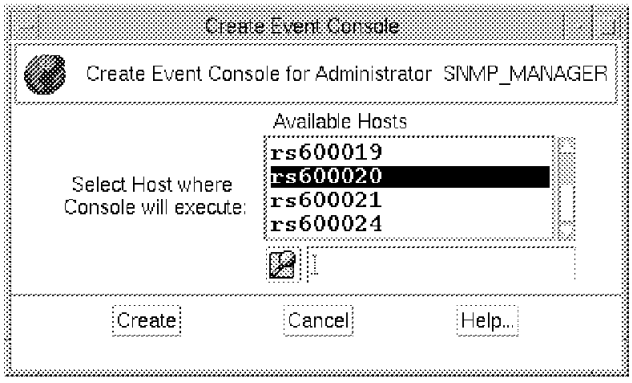


Figure 301. Where the Console Will Execute

The administrators desktop will now have an additional icon representing the console, as shown in Figure 302 on page 328. Notice that the user SNMP_MANAGER does not have the Event Server icon present on the desktop and therefore cannot gain access to the T/EC configuration.

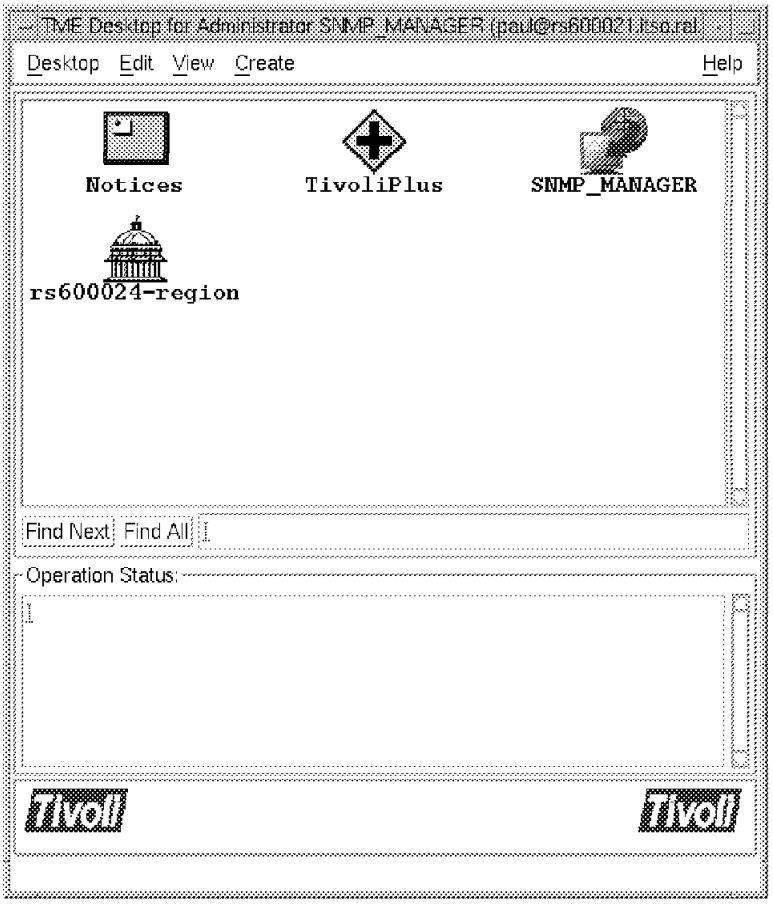


Figure 302. The SNMP_MANAGER Desktop

Creating Consoles from the Command Line: The T/EC consoles can also be created from the command line. The console for Paul was created by issuing the following command:

```
wcrtconsole @Paul
```

Our final configuration step was to copy the console icons from *all* of the administrators to the root desktop to allow access to the configuration. This stage is not specifically required but does make the configuration tasks quicker. The root desktop, with all three event console icons in place, is shown in Figure 303 on page 329

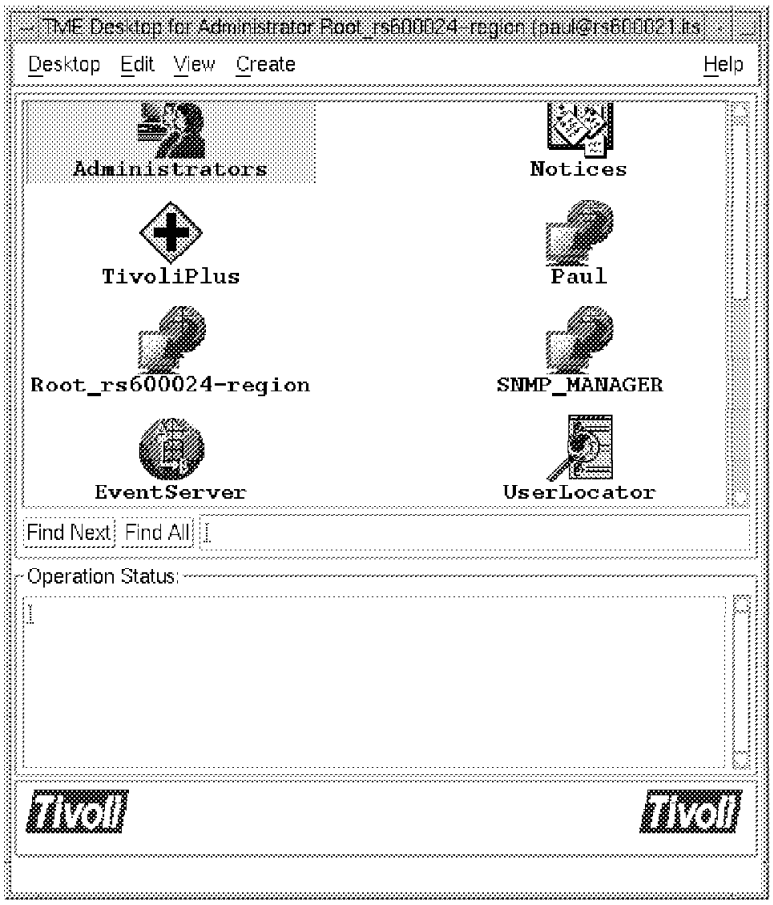


Figure 303. Root Desktop for T/EC Configuration

At this point we have created the consoles for each of the operators. However, nothing will appear on any of the console until we have configured the T/EC to receive events and deliver them to the consoles.

10.3.2 Initial Configuration of the T/EC Rulebase

At this point we do not want to get into the complex subject of creating T/EC rules (see Chapter 12, “More Advanced TME 10 Enterprise Console Customization” on page 383 for more details of this area). However, we do need to load a rulebase before the T/EC can operate. The reason for this is that the rulebase holds two types of information:

Event classes These define the structure of the arriving events. Each event adapter provides class definition in a *.baroc* file. You have to load the classes into the rulebase for each event adapter that you are going to use.

Event rules These define what actions are to be performed when specific events are received. The T/EC will operate successfully without any rules, so we do not have to worry about them at this stage. Some event adapters provide a set of default rules, but in most cases you will want to extend them to reflect your own local conditions.

In the example that follows, we will show the steps in creating a rulebase called *NetView* for our project environment.

There is one rulebase pre-installed in the T/EC, called Default. This rulebase is defined as read-only. The normal procedure is to define a new rulebase and copy the contents of the Default rulebase into it as a starting point. The new rulebase is then activated and any new classes and rules are added to it.

The event server should start automatically following its installation. You can see if it is active from the red arrow crossing the event server icon. If the arrow is not present, select the icon with the right mouse button and select **Start Server** from the menu.

To create the new rulebase:

1. Double-click on the **Event Server** icon from the desktop and select **Create** followed by **Rulebase** from the menu bar (see Figure 304).

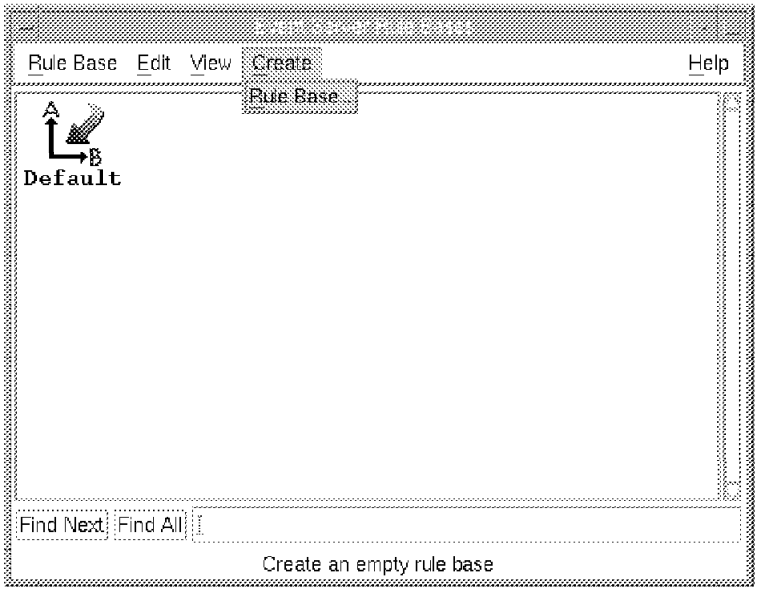


Figure 304. Create a New Rulebase

2. Enter the values shown in Figure 305. The directory that you specify here will be updated with a number of subdirectories containing the event classes and rules for the rulebase.

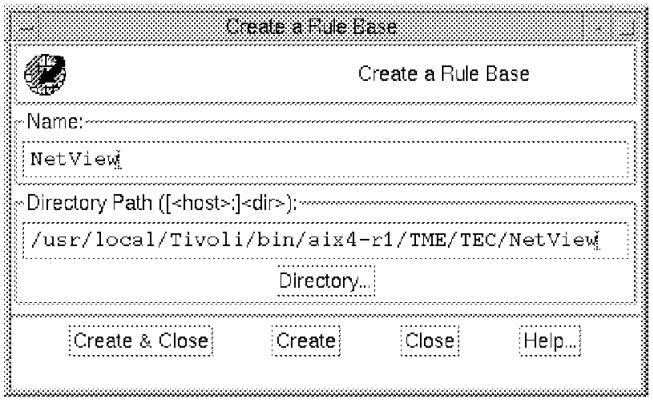


Figure 305. Define the NetView Rulebase

3. Copy the Default rulebase to the new rulebase. To do this, click on the **Default** icon and select **Copy** from the menu (see Figure 306 on page 331). Enter the fields as shown in Figure 307 on page 331.

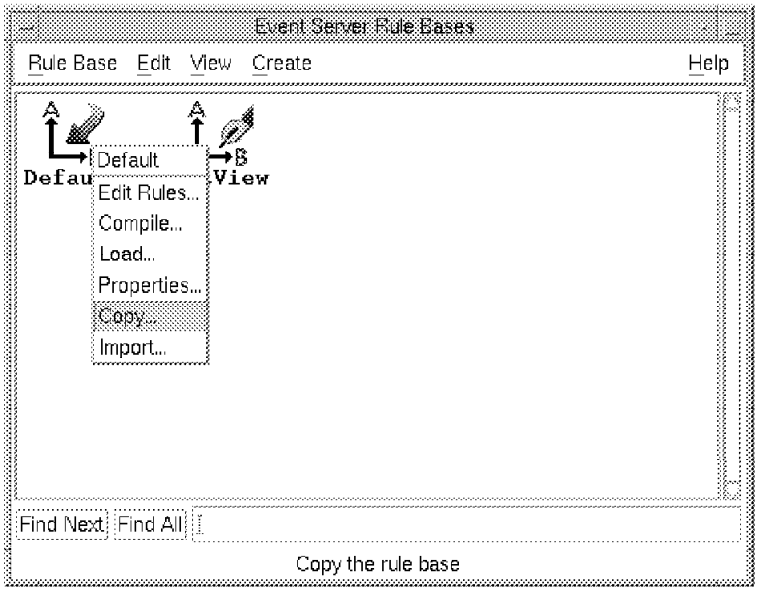


Figure 306. Copying the Rulebase

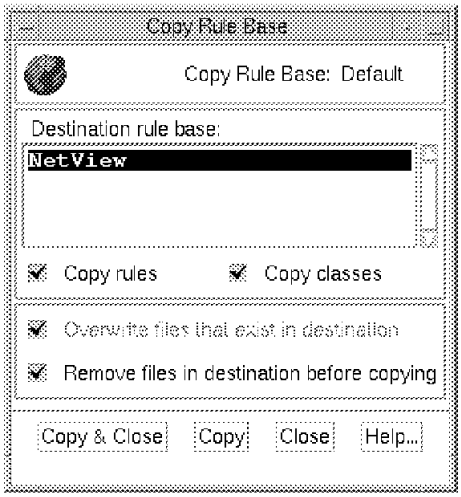


Figure 307. Defining the Target of the Copy Operation

In our case we copied all classes and rules into the NetView rulebase.

The final stage is to activate this new rulebase, by clicking on the new icon with the right mouse button and selecting **Load** from the menu (see Figure 308 on page 332). The red arrow will move from the Default icon to the NetView icon.

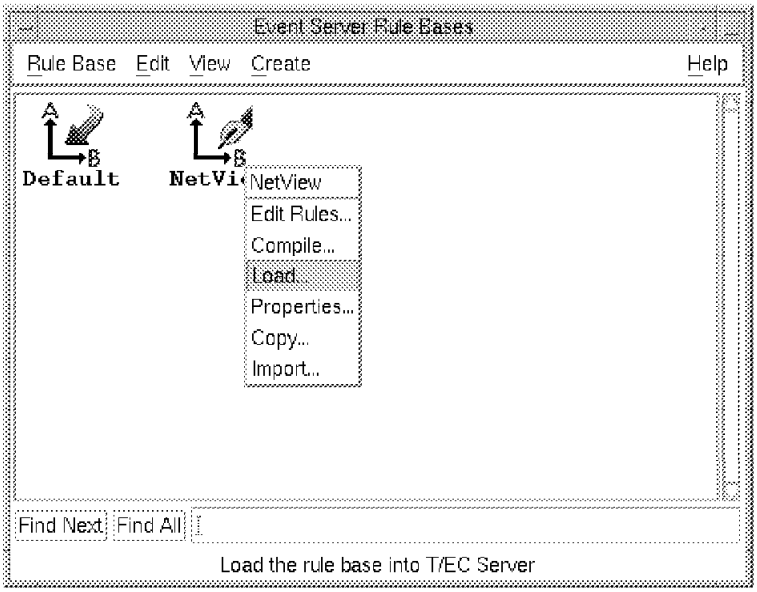


Figure 308. Activate the NetView Rulebase

When you load the rulebase you will be prompted whether to load it immediately or to wait until the event server is restarted. Figure 309 shows this dialog. Note, however, that if you are reloading the rulebase after having defined some new event classes, you will have to restart the event server. In this case, the rulebase we are loading is identical to the previous Default rulebase, so we select **Load and Activate**.

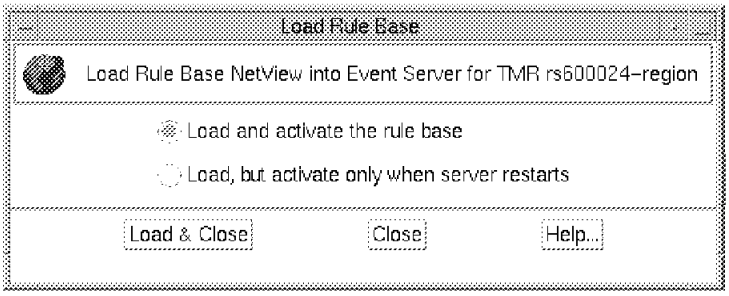


Figure 309. Activate the Rule Base

10.3.3 Importing Event Classes

The NetView rulebase is now active. The next step is to install some event adapters, so that the T/EC has something to work with. This procedure varies from one adapter to another, depending on the capabilities of the adapter code. We have described the installation and use of a number of different event adapters in Chapter 11, “Tivoli/Enterprise Console Adapters” on page 355.

For the purpose of the following examples we will configure one adapter; the NetView *nvserverd* adapter. The nvserverd adapter provides a file called nvserverd.baroc. This file contains the class definitions for the nvserverd events. To tell the T/EC engine to process these events we need to import these classes into the current NetView rulebase.

To import the nvserverd classes we performed the following steps from the root Tivoli desktop:

1. Click on the **Event Server** icon with the right mouse button and select **Rule bases** (see Figure 310).

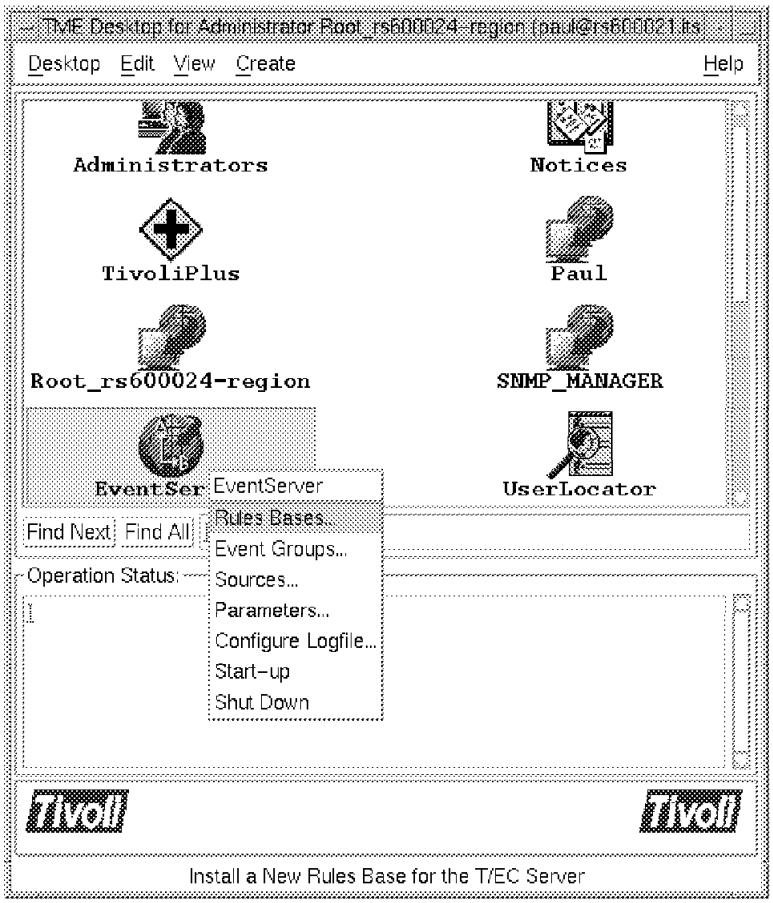


Figure 310. Open the Event Server Icon

2. Click on the active rulebase and select **Import** (see Figure 311 on page 334).

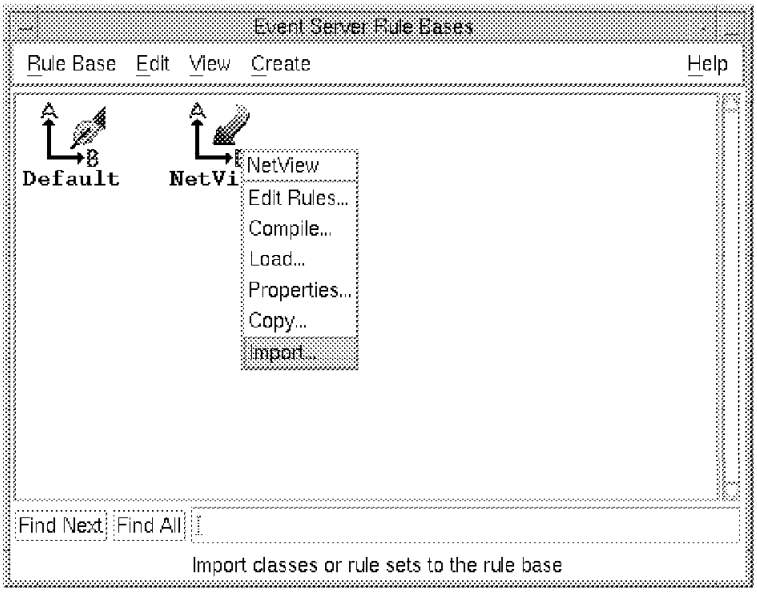


Figure 311. Import a New Class Definition

The nvserverd.baroc file was imported by filling in the fields shown in Figure 312 on page 335. These fields determine if the import is a ruleset or a class definition. In our case we are importing a .baroc file, which is a class definition.

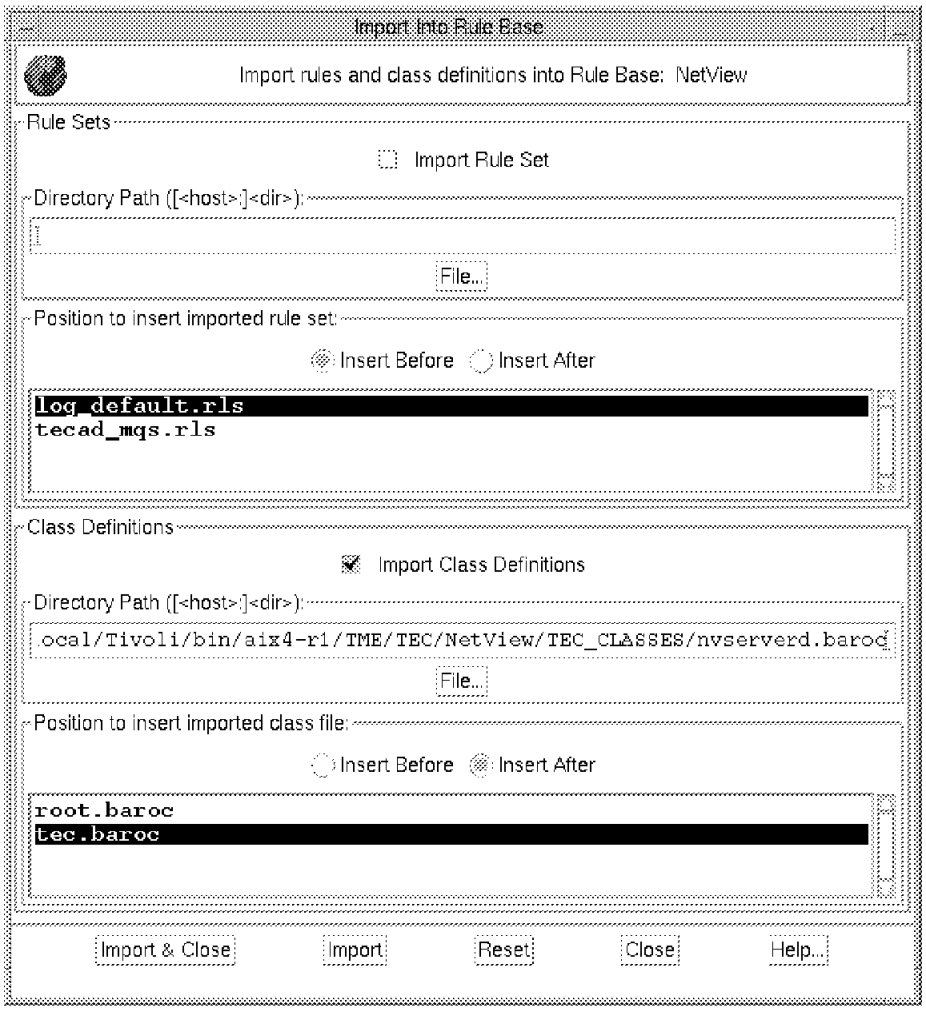


Figure 312. Import the Class Definition File nvserverd.baroc

The NetView rulebase must now be compiled by clicking on the **NetView** rulebase icon with the right mouse button and selecting **Compile** from the menu. Then the rulebase must be reloaded in and finally the event server must be stopped and restarted to activate the new class.

Create the NetView Rulebase from the Command Line: All of the functions for creating the new rulebase can also be executed from the command line, as follows:

1. Create a new rule base called NetView:

   ```
   wcrtrb -d /usr/local/Tivoli/bin/aix4-r1/TME/TEC/NetView NetView
   ```

2. List all the configured rulebases:

   ```
   wlsrb -d
   ```

3. Copy the existing Default rulebase into the NetView rulebase:

   ```
   wcprb Default NetView
   ```

4. Check the current loaded rulebase:

   ```
   wlscurrb
   ```

5. Import the nvserverd class definitions into the NetView rulebase:

   ```
   wimprbclass /<path>/nvserverd.baroc NetView
   ```

6. Compile the NetView rulebase:

   ```
   wcomprules NetView
   ```

7. Load the NetView rulebase:

   ```
   wloadrb -u NetView
   ```

8. Stop and restart the event server:

   ```
   wstopesvr
   wstartesvr
   ```

If you want to know more about the operation of these commands, look at the appropriate man pages.

10.3.4 Defining T/EC Groups and Sources

So far we have set up the event server to receive specific types of events and we have also defined some administrators who can start event consoles. The final piece of the puzzle is to define the T/EC output: which events each administrator will receive. The way you do this is by means of *event sources* and *event groups*.

Event sources define the type of adapter that events come from. Event groups use filters to define what type of event each console will show. The process below creates a number of groups for the different types of adapters we have running.

1. Select the **Event Server** icon using the right mouse button and then select **Event Groups** from the menu.
2. Select **Event Group** followed by **New** from the menu bar (see Figure 313).



Figure 313. Assigning New Event Groups

The source definition was added by selecting **Sources...** from the Event Server icon. The new source was called NetView Events. The configration is shown in Figure 314 on page 337.

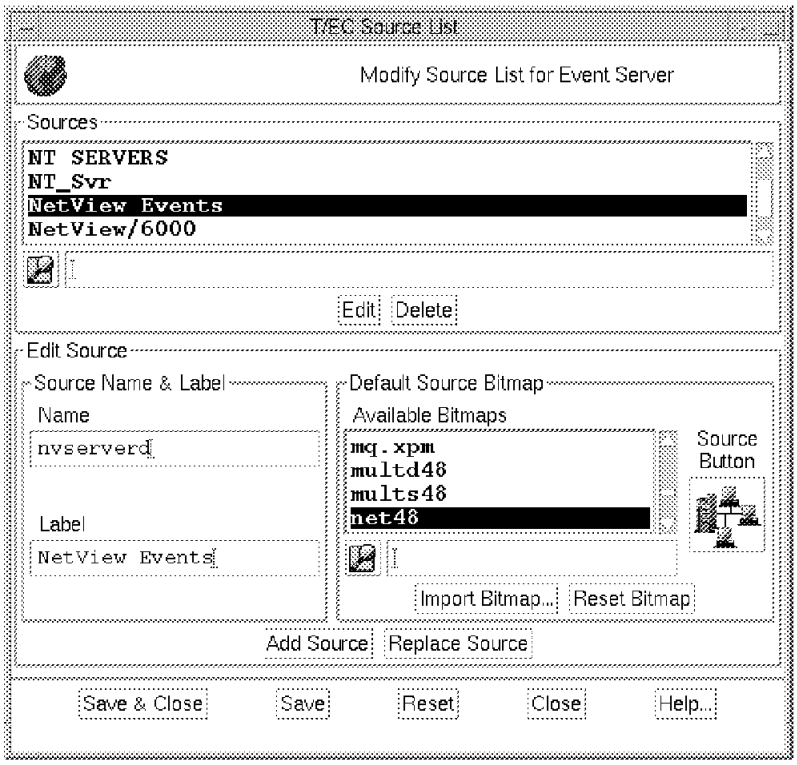


Figure 314. Assigning a New Source

3. Finally select **Save & Close**.

For each event group you have to specify a name and a bitmap for the icon that will represent the group, plus a set of filters that will control which events get passed to the console. We created a number of event groups, the example shown in Figure 315 shows the definition for the NetView_Events group.



Figure 315. Assigning New Event Groups

Note: The bitmap associated with the icon can be imported. We used a number of bitmaps. The bitmaps must be in xpm format.

The Source Type represents one of the *slots* in the baroc message format that is passed by the event adapter. In this case the source type of nvserverd is the

NetView adapter described in Chapter 11, “Tivoli/Enterprise Console Adapters” on page 355.

We created three event groups, each with different filters reflecting our three operator roles. Figure 316 lists them.

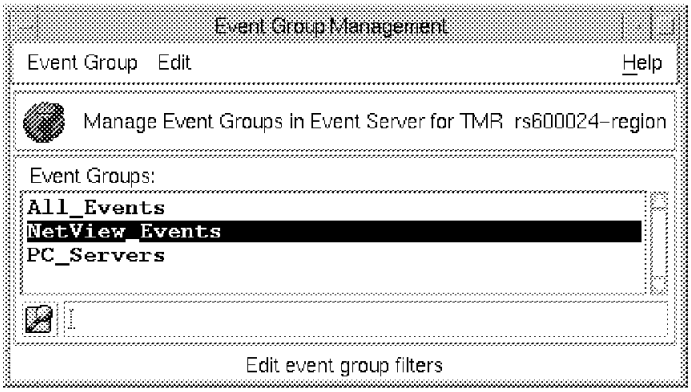


Figure 316. The Event Filter Definitions

10.3.4.1 Assigning the Groups to the Consoles

The three event groups can now be associated with the three consoles that we created earlier. The associations are as follows:

- Root_rs600024-region receives All_Events
- Paul receives PC_Servers and NetView_Events
- SNMP_MANAGER receives NetView_Events

For each console, we assigned the event groups by selecting the following:

1. From the icon menu for each administrator’s event console, select **Assign Event Groups** (see Figure 317 on page 339).

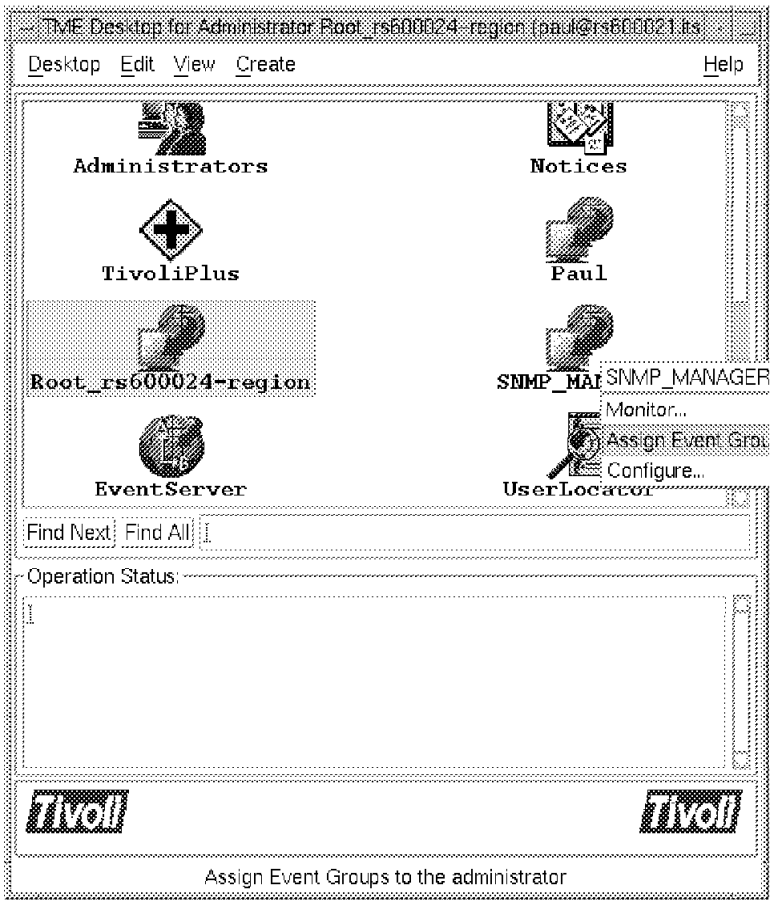


Figure 317. T/EC Event Group Selection

2. Move the appropriate group(s) into the assigned event groups list and assign roles to be associated with them. In our case we selected roles of admin and user (see Figure 318).
3. Select **Set & Close**.

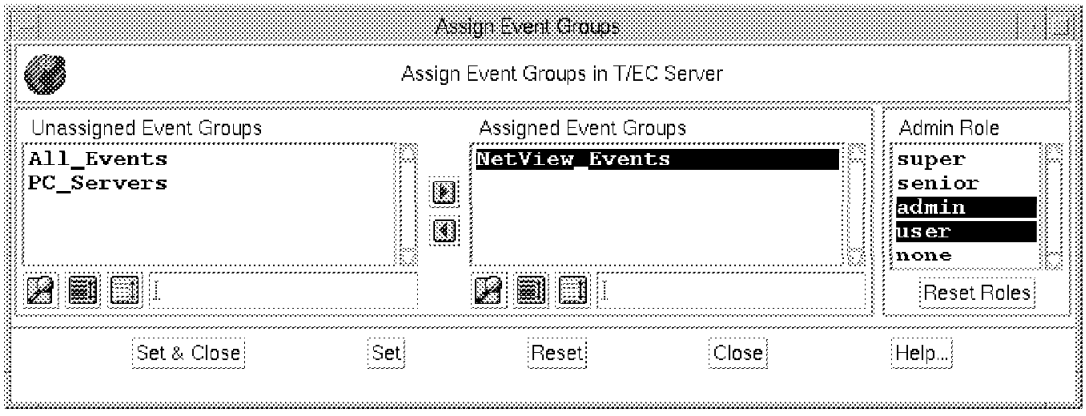


Figure 318. Assign Event Groups to SNMP_MANAGER Console

The T/EC console for SNMP_MANAGER should now be receiving events.

Configuring Groups from the Command Line

- First, create the event group and filter:

  ```
  wcrteg NetView_SNMP_Events
  ```

- Next, the filter must be defined:

  ```
  waddegflt -s nvserverd NetView_SNMP_Events
  ```

- Create a new console for the AIX user netview_man: and assign event filter NetView_SNMP_Events to it:

  ```
  wcrtconsole @SNMP_MANAGER
  wassigneg @SNMP_MANAGER NetView_SNMP_Events admin user
  ```

10.3.5 Configuring T/EC Server Parameters

There are a number of tunable parameters that will affect the number of events that will appear in the T/EC consoles and be retained in the server database. Normally, you can leave these values to default until you gain some experience using the T/EC. However, we will briefly review them here.

To amend the TEC server parameters click on the **Event Server** icon on the TME desktop with the right mouse button and select **Parameters**. You can specify how long events of different types are retained, and how big the event caches will be. Figure 319 shows the full set of options.

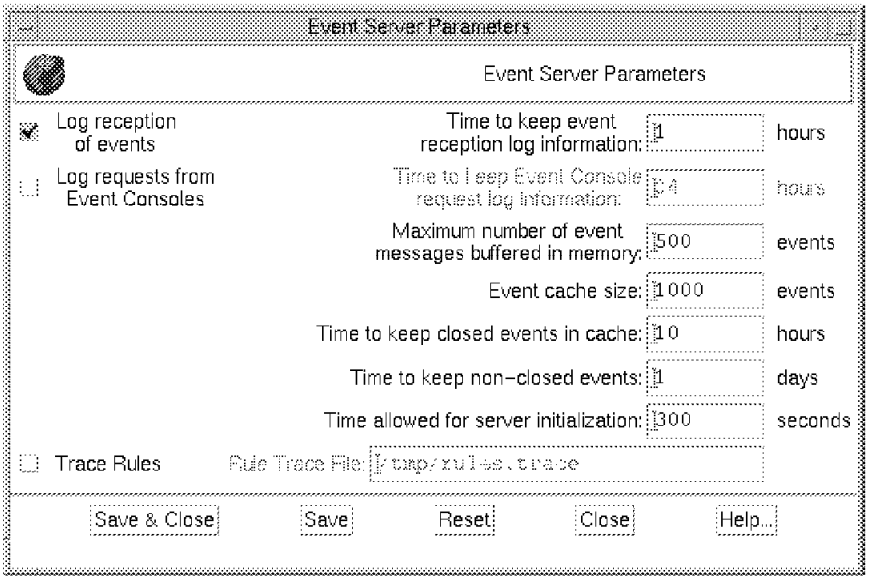


Figure 319. Setting T/EC Server Parameters

In addition to controlling how and when events are processed by the server, each user can set controls for their individual console. You set these limits by selecting **Configure** from the console interface menu bar. The settings for console Paul are shown in Figure 320 on page 341.

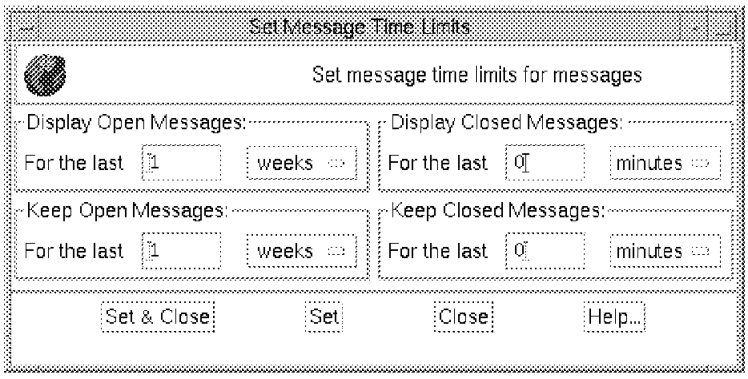


Figure 320. T/EC Console Message Limit Controls

The controls in this display are of two types. The display options control the number of messages that are loaded when the console is started. The keep options control what messages are retained in the console while it is running.

10.4 Using the Tivoli Enterprise Console Display

To start the console, log in as one of the administrators and double-click on the event console icon (it has the same name as the administrator, so if your user ID is SNMP_MANAGER the event console label would also be SNMP_MANAGER).

Two windows will appear: the Source Group and Event Group windows. Each of these contains a series of icons representing the events from a specific group or source. For example, Figure 321 shows the single event group for user SNMP_MANAGER. Alongside the icon is an indicator if the most severe event, in this case severity *FATAL*.

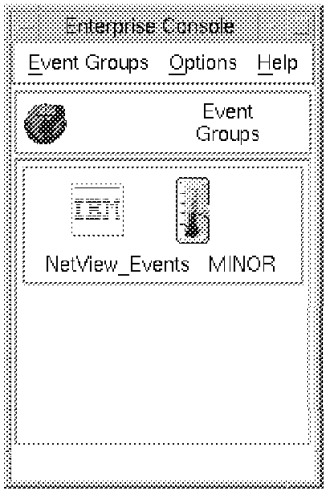


Figure 321. The T/EC Event Console Group Display

If you click on the group icon you will see the T/EC console with just the events from that group displayed, as shown in Figure 322 on page 342 for the NetView group.

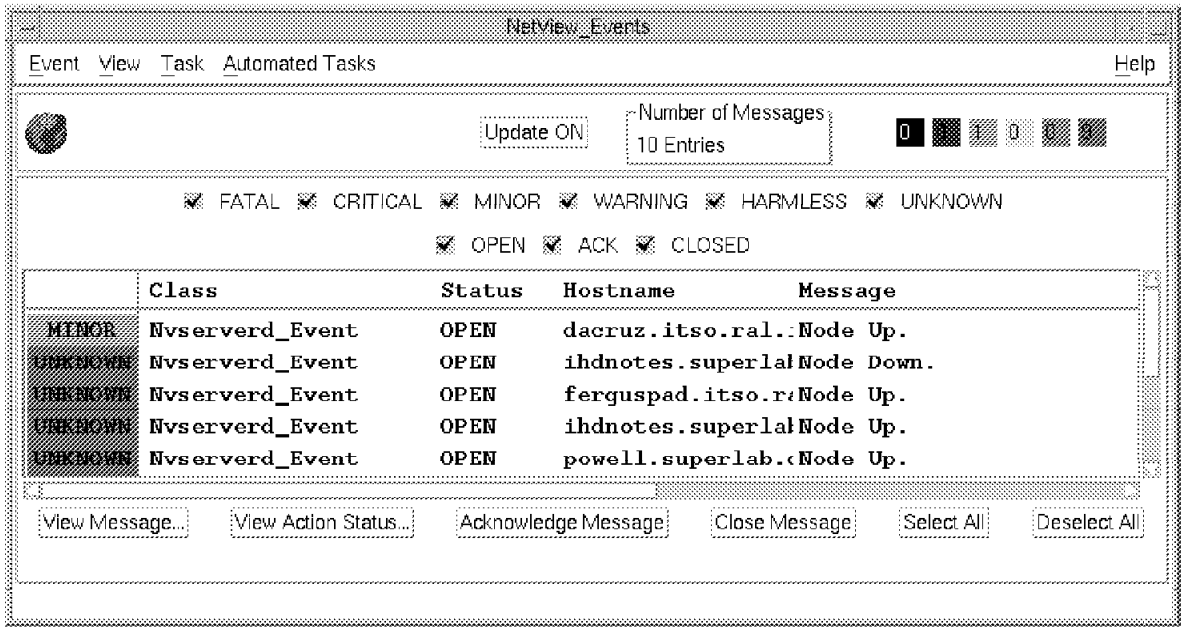


Figure 322. The T/EC Event Display

By selecting an event and then selecting **View Message** more details of the message are shown (see Figure 323). In fact, the entries in this display are the message slots from the BAROC format.



Figure 323. View Message Details

By comparison, the administrator Paul has two groups assigned, so he has two group icons to select from (see Figure 324 on page 343). This means that Paul

can start two console event displays, one showing NetView events, the other showing all PC Networking (Novell, NT Server) events.

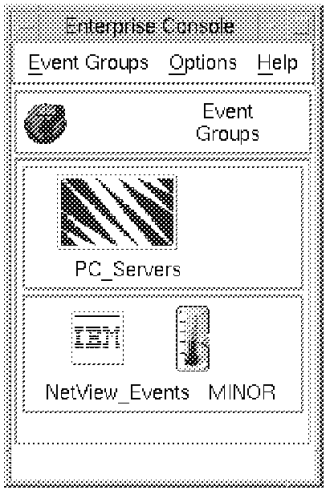


Figure 324. Group Display for Paul

10.4.1 Tuning the T/EC Interface

The appearance of the events can be modified using the T/EC console menus. You can alter the sort order of the events and also the fields that are displayed on the panel. We sorted the events by date and altered the order in which message fields are displayed. Figure 325 and Figure 326 on page 344 show the modification dialogs, and Figure 327 on page 344 shows the end result.

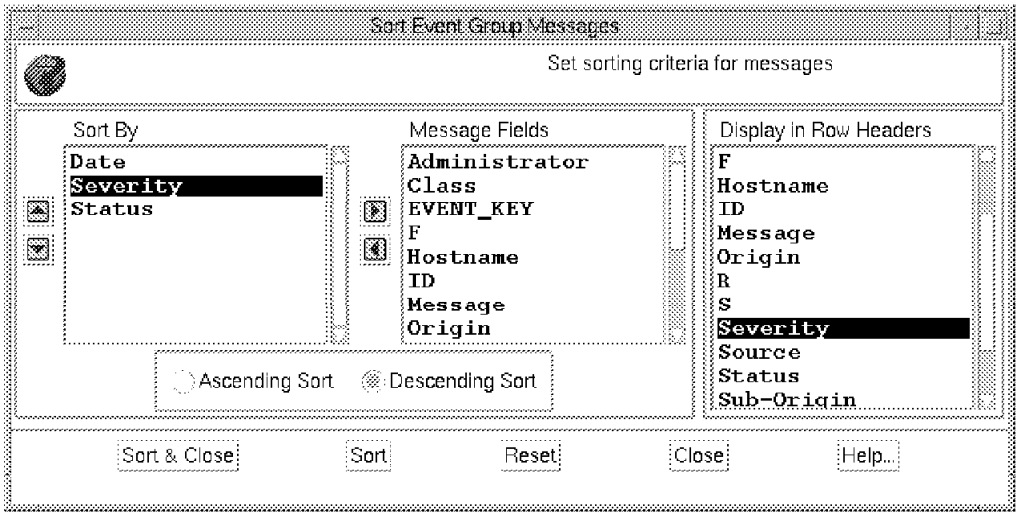


Figure 325. Altering the Sort Order for an Event Console

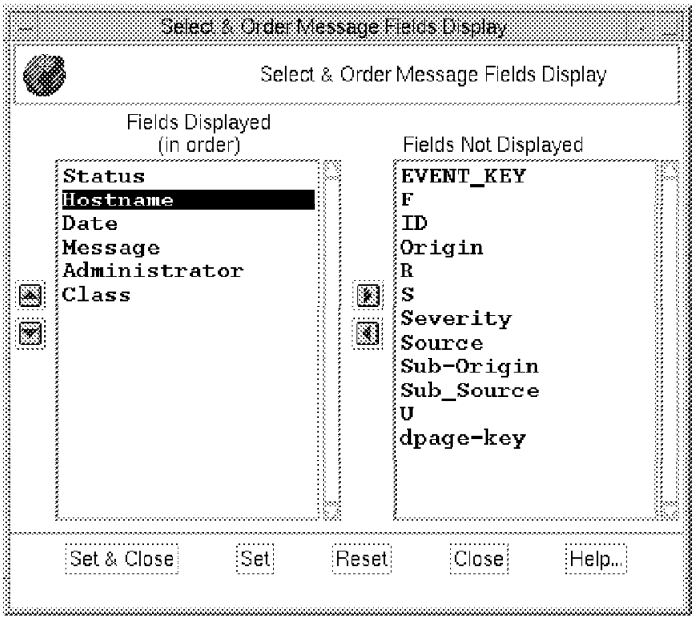


Figure 326. Altering the Display Fields for an Event Console

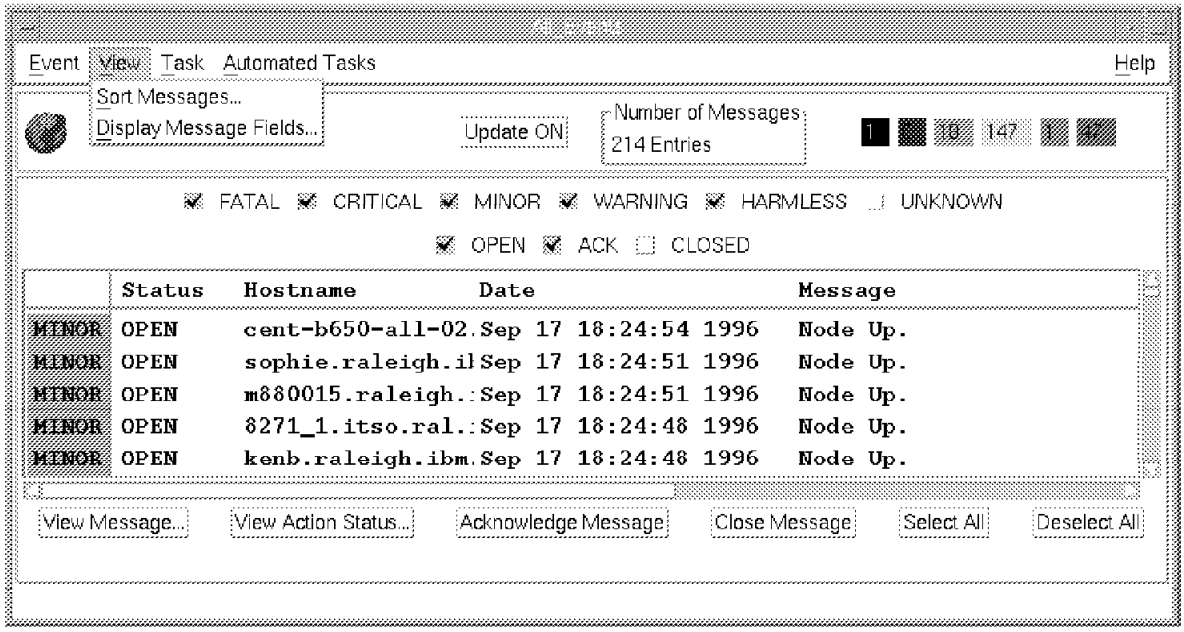


Figure 327. The Customised Events Display

10.4.2 T/EC Tasks

The T/EC provides the user with the ability to execute tasks from the T/EC console display. The tasks can be executed on any managed node or on the specific node from which the event was generated. The examples below show how to run a task against a particular event and automatically invoke a task on receipt of a specific Node Down event.

When the T/EC is installed a default set of tasks are included. These tasks are contained within the T/EC task library. The task library is located within the

TEC25Region policy region. You can also use the desktop navigator to locate them, if you want. (see Figure 328 on page 345).

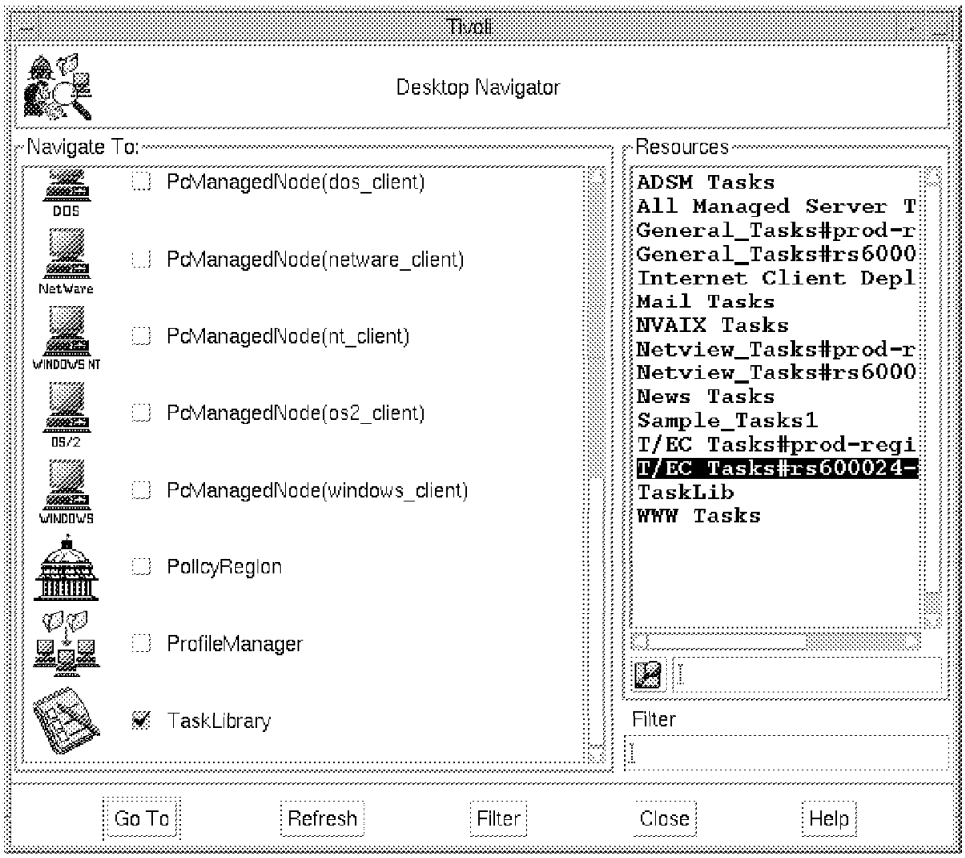


Figure 328. Navigator Window

The tasks in the standard T/EC task library are shown in Figure 329 on page 346.

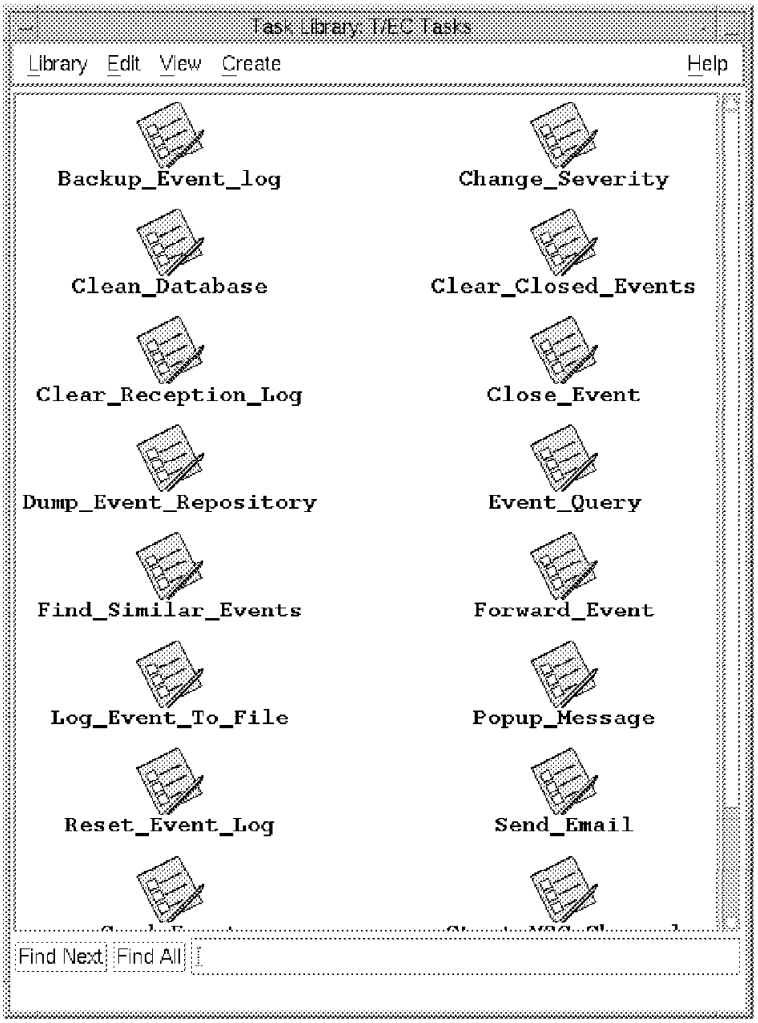


Figure 329. T/EC Task Definitions

10.4.3 Running a Task Against a Particular Event

Here we show how to run a task from the T/EC console. The task we want to execute is a predefined T/EC task that will raise a pop-up window with a message on another administrator's window. The message text is held in a file on machine rs600020.

To run the task:

1. Select **Task** followed by **Execute** from the T/EC events GUI.
2. Double-click on T/EC Tasks to show the tasks contained in the default library.
3. Select **Popup Message on Desktop** from the Tasks window.
4. Click on **Select Task**.
5. Enter the fields in the resulting panel.
6. Select **Execute**.

The message will be sent from your administrator ID to the target administrator (in our case, Root_rs600024-region) as shown in Figure 330 on page 347.

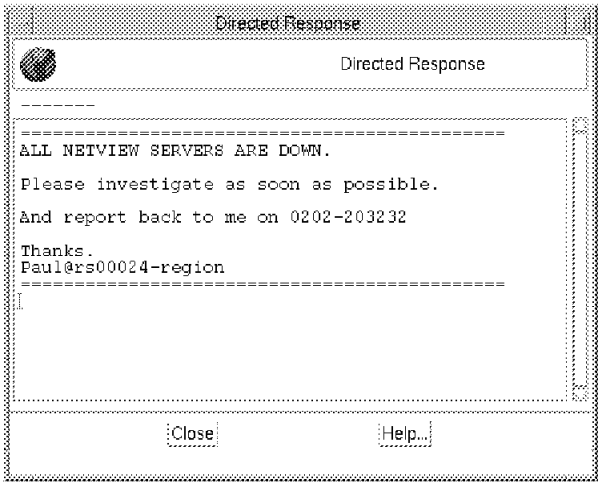


Figure 330. The Message Appearing on the Console

The next stage is to automate the task when a specific event is received by the T/EC server.

10.4.4 Adding an Automated Task

Here we added an automated task to the T/EC console called NetView_Server_Down. This automated task raises the severity level of the Node Down for the machine rs600021 only.

To create the automated task, we did the following:

- From the T/EC event display select **Automated Tasks** followed by **New**.
- Select the event class **OV_Node_Down** (see Figure 331 on page 348).
- Click on **Select & Close**.
- Click on **Edit Criteria**.
- Click on **Hostname**.
- Click on the right hand arrow icon.
- Add the hostname rs600021.itso.ral.ibm.com to the hostname field (see Figure 331 on page 348).

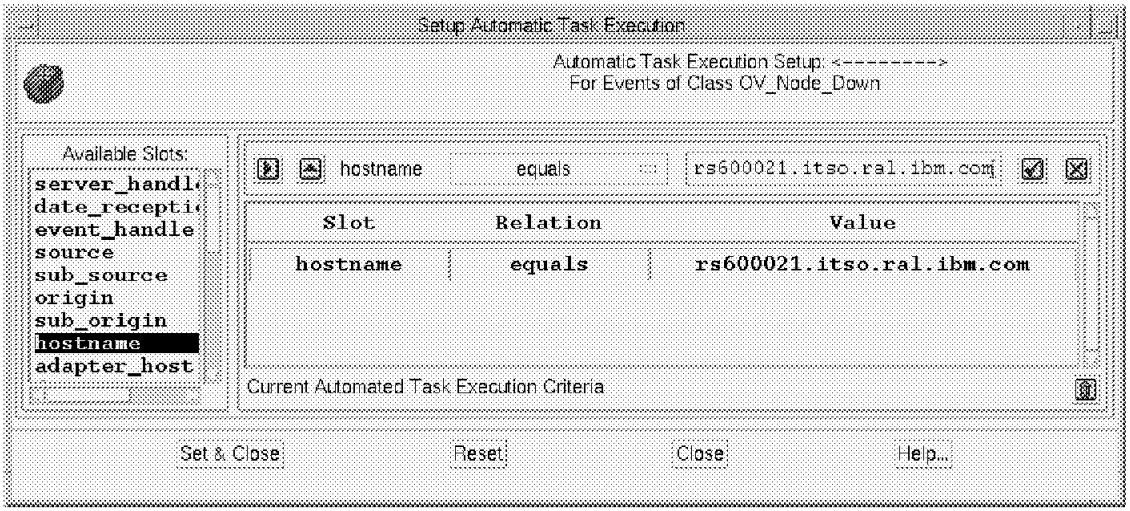


Figure 331. Edit Criteria Fields

- Click on the check mark icon and then select **Set & Close**.

Next we added the task to run when this criteria is met.

- Select **Add Task** and select the task we want to be automatically performed (in this case, Change severity of event).
- Select **Add & Close** and **Save**.

The final panel is shown in Figure 332.

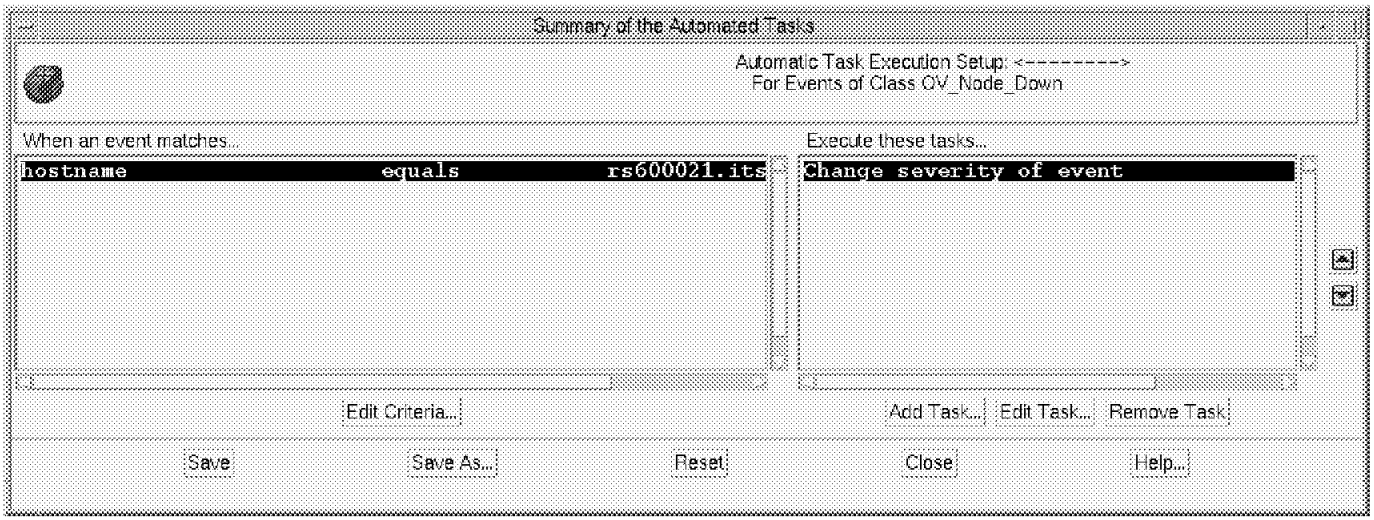


Figure 332. Final Automated Task Configuration

From this point on, any Node Down event for machine rs600021 receives a severity of *CRITICAL*.

10.4.5 Creating a New T/EC Task

New T/EC tasks can be added to the T/EC library and executed from the console. For this example, we create a new task to call up a Web page for the NetFinity server. The NetFinity server called WTR05119 was running TME 10 NetFinity Version 4.

The command we want to execute is:

```
netscape -remote "openURL(http://WTR05119.itso.ral.ibm.com:411/main)"
```

First, we create a shell script called /u/paul/TEC_NetFinity.sh that contains the netscape command.

Next, we add the task definition to the T/EC Task library by issuing the command:

```
wcrttask -t TEC_NetFinity -l "T/EC Tasks#rs600024-region" \
-r user -c "Call NetFinity URL" -i default \
rs600024 /u/paul/TEC_NetFinity.sh
```

The new task will be displayed as an additional T/EC task as shown in Figure 333.

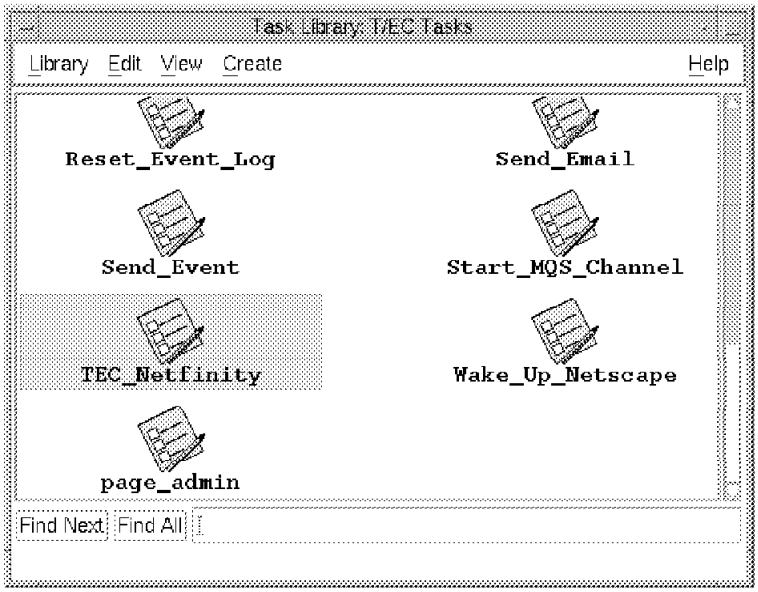


Figure 333. The New Task TEC_NetFinity

To see if the task has been added to the correct task library enter the command:

```
wlstlib "T/EC Tasks"
```

To see the properties of a particular task issue the command:

```
wgettask "TEC_NetFinity" "T/EC Tasks"
```

The output from this command is shown below.

```
Task Name               TEC_NetFinity
User Name
Group Name
Task ACL                user
Supported Platforms
default         <install-dir>/generic_unix/TAS/TASK_LIBRARY/bin/1131
 17/T_EC_Tasks_T_msxmknea
Task Comments
Task Name            : T/EC Tasks/TEC_NetFinity
Task Created         : Wed Sep 11 14:28:06 1996
Task Created By      : root@rs600024.itso.ral.ibm.com
Task Files
default          rs600024
/usr/local/Tivoli/bin/generic_unix/TAS/TASK_LIBRARY/bin/1131960017
 /TEC_NetFinity.sh
Distribution Mode    : ALI
Call NetFinity URL
```

Finally, we test the task from the command line by issuing the command:

```
wruntask -l "T/EC Tasks" -t "TEC_NetFinity" -h rs600024
```

The Web interface was shown as in Figure 334 on page 351.

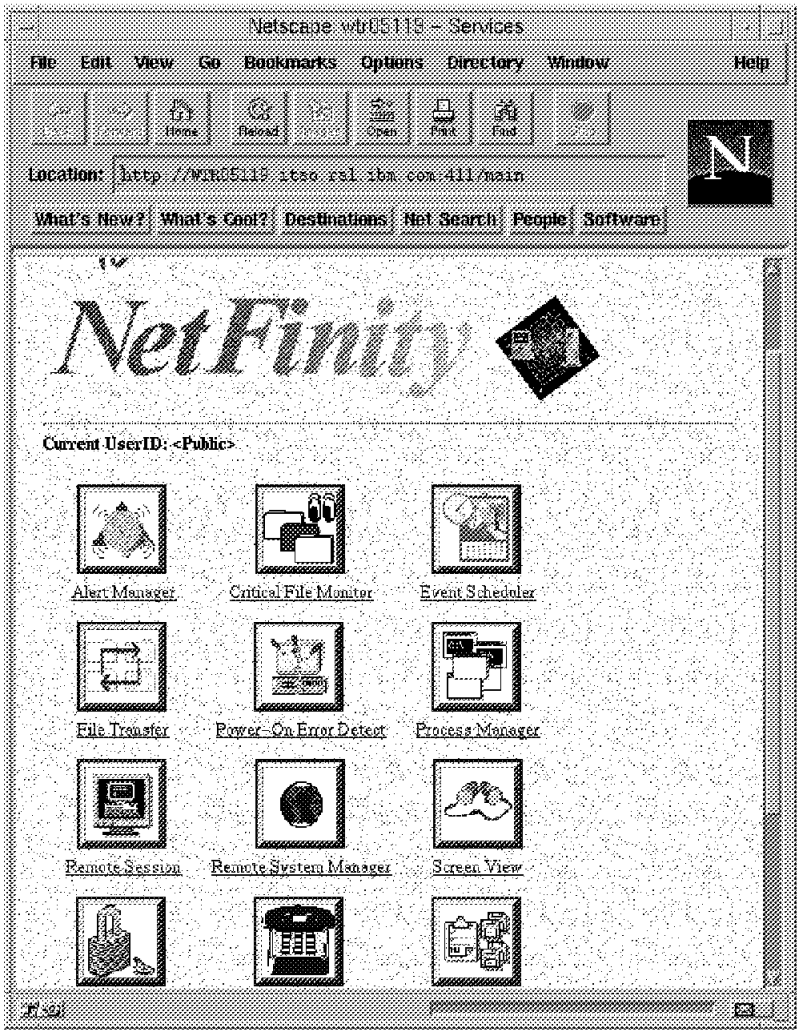


Figure 334. NetFinity Web Page

The new task will now appear as one of the available tasks to be performed by the T/EC from the GUI.

If there is an error recorded during the execution of the shell script then any modifications made to the script requires the task to be deleted from the library and re-created.

The TEC_NetFinity task can now be accessed from the Tivoli GUI. (see Figure 335 on page 352).

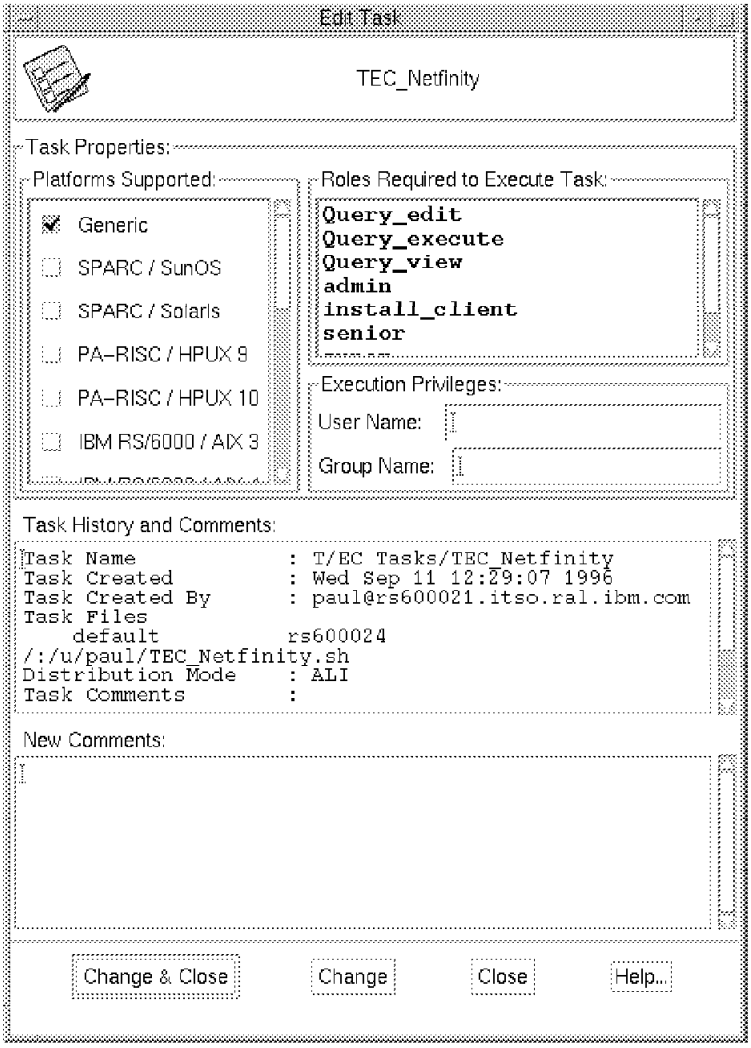


Figure 335. The TEC_NetFinity Task Definition

This new task can be selected from the T/EC Text window as shown in Figure 336 on page 353.

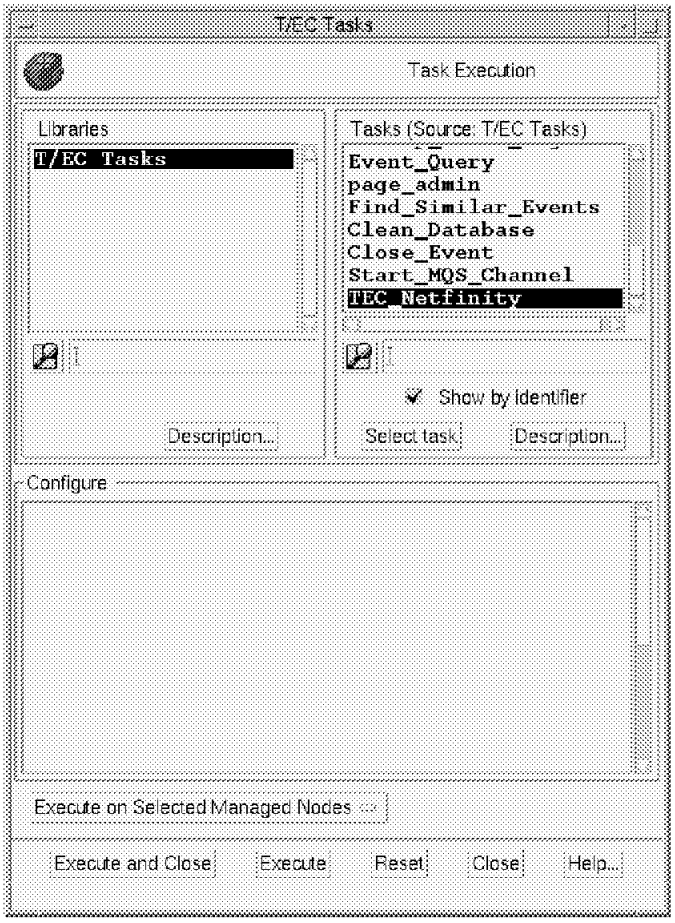


Figure 336. The TEC_NetFinity Task

Notice we had to click on **Show By Identifier** for this panel. Otherwise, the task would have shown up with an Upgraded Task label.

For TME region rs600024-region, the task scripts were held in the directory:

```
/usr/local/Tivoli/bin/generic_unix/TAS/TASK_LIBRARY/bin/1131960017
```

All of the available task scripts are held on the TMR server. In our case this was rs600020.

10.5 Useful Commands

There are a number of commands that are useful for diagnosing problems with T/EC processing.

Checking for Event Reception: On the T/EC server, execute the following command to display which events have been received, and their status within the rule engine:

```
wtdumprl -o DESC | more
```

The `-o DESC` flag causes the results to be sorted in descending order, so that the most recently-arrived events are at the top. The kind of output that the command gives is shown in Figure 337 on page 354. In this case you can see

that an event has arrived for which the class has not yet been imported into the T/EC server (see 10.3.3, “Importing Event Classes” on page 332).

```
### EVENT ###
Nvserverd_Event;source=nvserverd;sub_source=N;origin=9.24.104.160;hostname=wtrpr
t02.itso.ral.ibm.com;adapter_host=rs600019;date="08/20/96 14:20:55";status=OPEN;
severity=UNKNOWN;msg="Node Up. ";nv_enterprise=netView6000;nv_generic=6;nv_speci
fic=58916864;nv_var1=2;nv_var2="wtrprt02.itso.ral.ibm.com";nv_var3="Node Up.";nv
_var4="840568855 358";nv_var5="openview";END

### END EVENT ###
PARSING_FAILED 'Line 1: Class Nvserverd_Event undefined'
```

Figure 337. Example of wtdumprl Output

Displaying T/EC Database Statistics: There are particularly useful commands for diagnosing problems with the RDBMS component of the T/EC. To see if the database server is running or not, enter the following:

```
wtdbstat
```

You can also look for the data_server daemon process using the ps command.

To check the amount of space being used by the database tables, issue the following command:

```
wtdbspace
```

The result of this command is a display similar to Figure 338. If you see free space in any resource drop to less than 2-3%, you should perform some housekeeping. Otherwise, the T/EC may behave unpredictably.

```
Summary:
Resource (Approximate)         Total       Used     Free     %Used     %Free
-------------------------------------------------------------------------------
Data Space:                 30.00 MB    5.07 MB   24.93 MB   16.91     83.09
Log Space:                  30.00 MB   28.44 MB    1.56 MB   94.79      5.21
Event Space:                   26664       4510      22154   16.91     83.09
```

Figure 338. Result of the wtdbspace Command

T/EC Database Housekeeping: You can clear events from the T/EC database and from the transaction log using the following commands. To clear the database, enter:

```
wtdbclear -el -t 0
```

To backup and clear the transaction log:

```
wtdbbackup
```

Normally, if you have allocated enough space for the database tables, you should not need to issue wtdbclear because old records will be removed according to the parameters you set for the T/EC server (see Figure 319 on page 340). However, you should schedule the wtdbbackup command at regular intervals.

Displaying Rulebase Information: To list the rule base directories:

```
wlsrb -d
```

Error Log Files: Some filenames containing errors are listed below:

- /tmp/tec-rdbms.sinst
- /tmp/TEC_DB_BIN_after.error

Chapter 11. Tivoli/Enterprise Console Adapters

As far as the Tivoli/Enterprise Console is concerned an event is an unsolicited, structured message from any node. The function that captures these messages is called an *event adapter* and it may exist on any TCP/IP-connected system, not only on TME Managed Nodes or PC Managed Nodes. The messages are in a special structured format named *baroc* (pronounced ″baroque″ but without the wigs and frilly neckwear), an abbreviation of Basic Representation of Objects in C).

There are a number of different types of T/EC event adapters that may be loaded on remote machines. They are responsible for generating the events and sending them to the T/EC console. The adapters are also resposible for any local filtering of events that may be required to help reduce network traffic and event server load.

Baroc is a simple event class structure that allows you to separate the important elements of a message into a number of individual pieces of information called *slots*. Each event is defined as a member of a class. The baroc structure then allows you to create subclasses of the master class. Subclasses inherit slot definitions from the higher-level classes. This means that you only need to define the event slots that are common to all events (such as date and time, severity, and origin). This will become more clear as we go into some baroc examples.

In this chapter we describe the installation and operation of a number of event adapters, namely:

- Logfile adapter
- NetView for AIX nvserverd
- NetView/Openview trapd
- Sentry Integration with the T/EC Server
- Windows NT Adapter
- Generating events from the command line

Before look at the adapters, we briefly describe the elements that are common to most of them, namely the installation process and some of the configuration files.

11.1 Installing the Event Adapters

Most of the event adapters we cover are provided on the T/EC CD. They are installed from the Tivoli desktop like any other TME product, by selecting Desktop followed by Install product. This process is described in detail in 5.2, “Installation” on page 109.

© Copyright IBM Corp. 1996

11.2 Adapter Configuration Files

The operation of each adapter is controlled by a number of files. All of the files are used by the adapter directly (and must therefore reside on the source machine) with the exception of the baroc file. This file contains the event format to load into the event server. It may be located on either the source system or the T/EC server.

The event adapters are not consistent in where they place their configuration files, so we found it useful to practice some discipline when updating them. After each adapter was installed we made a copy of all the class definitions, rules and configuration files to an area on the T/EC server. We then made updates to this central repository and copied them to the remote systems. This prevented some confusion and frustration, since we knew exactly where to find the current copy.

The following list explains the contents of the common adapter configuration files. Every adapter has one or two of these files, but may not have all of them, depending on the function it performs. We have listed the file extensions here.

.baroc The class definitions required by the T/EC server to process the event. This file is imported into the T/EC server as a class definition.

.conf The main configuration parameter file for the adapter. This file tells the adapter where to find the event server, sets options such as maximum message size and also provides some filtering options.

.cds The class mapping file that resides on the remote machine. This file holds the information on how the baroc slots in the event will be derived from an original unstructured message. CDS stands for *Class Definition Statements*.

.fmt A simplified mapping file that is converted into a cds file by passing it through a filter program (logfile adapters).

.err This file contains the tracing options for some adapters; the options are to send debugging information to a file or panel.

.oid For SNMP, NetView and Openview adapters, this contains mappings from SNMP MIB object IDs to plain text.

11.3 The Logfile Adapter

All operating systems and applications have some kind of logfile to document the running of programs and processes. Each entry in these files represents the confirmation of an activity, whether it was successful or not.

If you are looking for errors, reading logfiles is sometimes the only way to diagnose what happened. However it is not easy for you to get this information. First you have to determine the location and the filename of the logfile. In addition, the quality of the information that systems and applications place in logfiles varies enormously. You may therefore find yourself looking at a sequence of plain text messages, written one after another with no consideration for severity or meaning.

Some operating systems provide a general purpose logging mechanism, which system components and applications can use. One example is the UNIX syslogd daemon. With the T/EC logfile adapter it is possible to generate T/EC events

based on messages from syslogd, as well as any other logfiles on the system. The logfile adapter provides a set of baroc classes to represent these messages.

The logfile adapter checks the messages in the logfile using pattern matching. If the message matches a specific pattern entry, the adapter can then assign parts of the message to the fields of a particular event class. It may not be clear at this point why it is important to be able to do this. However, when we go on to consider T/EC rules in Chapter 12, "More Advanced TME 10 Enterprise Console Customization" on page 383, we will see that it is a prerequisite for the rule engine to be able to correlate events and start actions automatically.

Note that the T/EC logfile adapter is only available for UNIX systems. There is an equivalent adapter for monitoring logfile information on NT machines (see 11.8, "Windows NT Adapter" on page 379).

11.3.1 Installation of the T/EC Logfile Adapter

You should follow the normal TME install procedure to install the logfile adapter on a TME Managed Node (see 5.2, "Installation" on page 109). The only option that the installation gives you is to ask if you want the adapter automatically started when the system is rebooted. It is not necessary to import new class definitions and rules, since the installation process automatically imports them into the Default rulebase.

The logfile adapter can also be installed on clients without the Tivoli Management Environment installed. In this case it is not possible to install the logfile adapter from the TME desktop. You should also note that when the adapter is installed in this way it does not have the security benefits of using TME platform services to communicate with the T/EC server.

Install the non-TME version of the adapter manually using the following steps:

1. Create the installation directory for the adapter on the local node:

   ```
   mkdir /usr/tecad
   ```

2. Set up the environment variable in your login profile:

   ```
   export TECADHOME=/usr/tecad
   ```

3. Install the platform-specific adapter into your installation directory (the source directory depends on the particular operating system):

   ```
   tar -xvf /cdrom/ADAPTERS/AIX3-R2/LOGFILE.TAR
   ```

11.3.1.1 Prepare the T/EC Server to Process Events from the Logfile Adapter

The logfile adapter will work as soon as it has been installed, but the T/EC server must be prepared to receive the logfile events. We described this process in 10.3.2, "Initial Configuration of the T/EC Rulebase" on page 329; it is only necessary for the first installation of the adapter. In summary, the steps involved are as follows:

1. Create a new rulebase that will be your working rulebase.
2. Copy the Default rulebase into your working rulebase.
3. Import the class definition into the rule base. You only need to do this if you copied the Default rulebase *after* you installed the logfile adapter on a managed node. The class definition file is named tecad_logfile.baroc and it is found in directory /etc/Tivoli/tecad/etc. You can use the `wlsrbclass`

command to determine what event classes have been imported into your rulebase.

4. Compile the rulebase.
5. Load the rulebase.
6. Stop and restart the event server.

These steps can be performed either from the GUI or by using line commands, as described in “Create the NetView Rulebase from the Command Line” on page 335.

11.3.1.2 Prepare to Display Logfile Events in Event Consoles

Now that we have defined the logfile input for the T/EC server, we need to configure the output using three steps:

1. Define a new source type for the logfile events.
2. Create event group filters that select logfile events to be displayed.

 Figure 339 shows the dialog for creating the logfile adapter source, and Figure 340 on page 359 shows the source being used in a filter to define an event group containing logfile adapter events. For a full description of this process, refer to 10.3.4, “Defining T/EC Groups and Sources” on page 336.

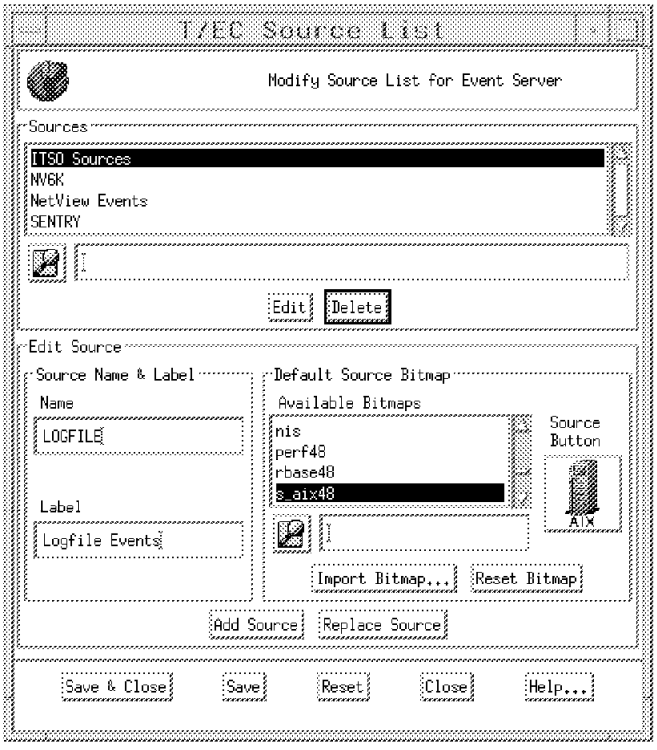


Figure 339. Defining the LOGFILE Source

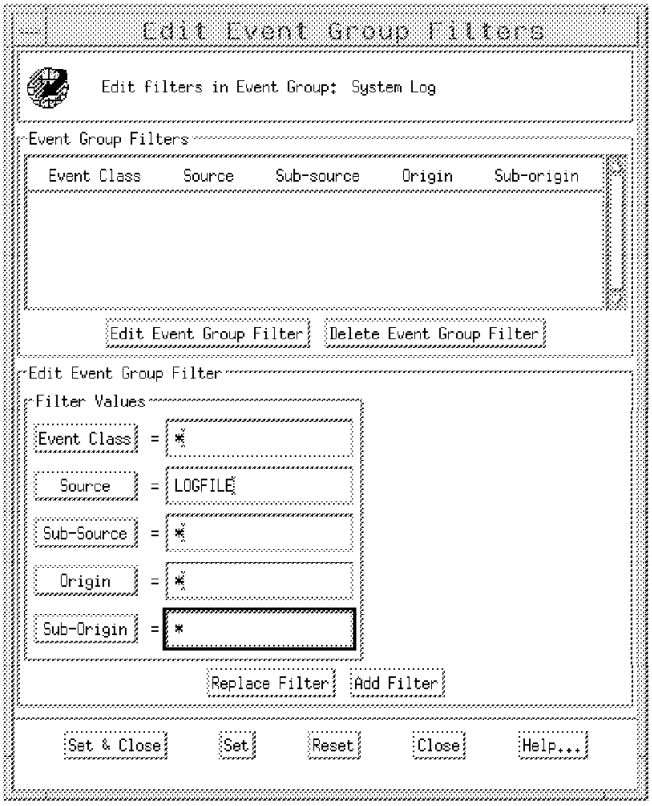


Figure 340. Define Filters for Event Group 'System Log'

3. Assign the event group to your event console.

 Figure 341 shows this operation. Refer to 10.3.4.1, "Assigning the Groups to the Consoles" on page 338 for a full description of the process. Do not forget to assign authorization roles when assigning groups. The administrator needs the user role to view events and the admin role to be able to close, acknowledge, and execute actions against events.

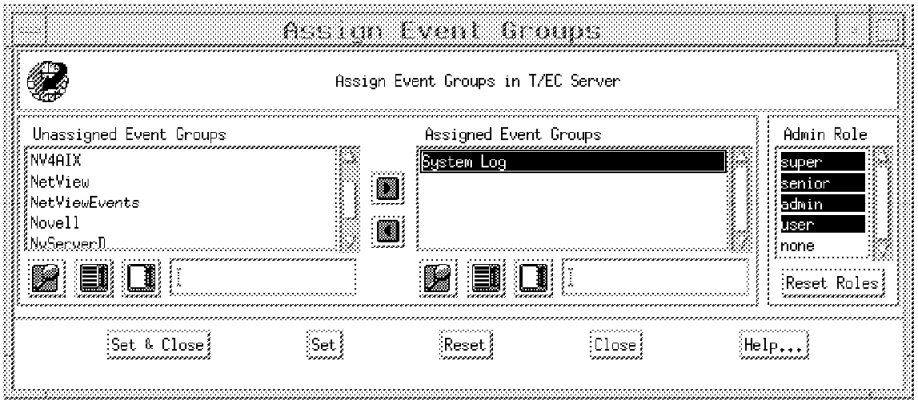


Figure 341. Assign Event Group 'System Log' to an Event Console

11.3.1.3 Testing the Installation

Having set up the logfile adapter we should now be able to send events to the event console. The su command (switch user) is a good way to test this functionality. This command writes a message to syslog, which the logfile adapter, by default, is configured to convert into a T/EC event. Change your user identity to any known user account using su and a new event should be displayed at your event console. If everything has been set up successfuly you will see a message similar to that shown in Figure 342.

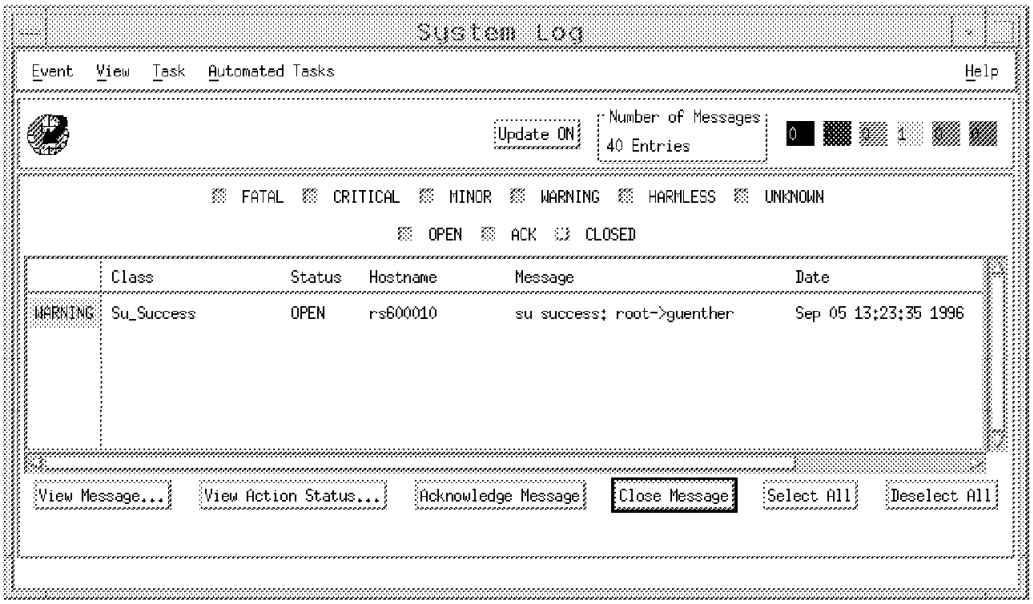


Figure 342. Su_Success Event Sent from Logfile Adapter

If you don't receive the event, the following sequence of checks may help you pinpoint the problem.

Table 13 (Page 1 of 2). T/EC Troubleshooting

Question	How to perform the check
Have you set up the Tivoli Environment?	Enter the command echo $BINDIR. If you get no output enter the command . /etc/Tivoli/setup_env.sh.
Is the object dispatcher running?	Enter the command odadmin. If you see the o_self: destination dispatcher unavailable message, this means that oserv is not running. Enter odadmin start to restart it. If it fails to start successfully, look for errors in $DBDIR/oservlog.
Can you send the event to T/EC manually?	Enter the command wpostemsg Su_Success LOGFILE. Normally this command gives no output. If instead you see the Error creating a connection to the Event Server message, you should check that the event server is running by issuing the command wstatesvr. If the events server *is* running, check that the event server host is correctly defined in the object database (below).

Table 13 (Page 2 of 2). T/EC Troubleshooting

Question	How to perform the check
Is the EventServer host correctly defined?	Enter the command `wlookup -r EventServer -a`. The response should be a something like this: `EventServer 1131960017.2.30#Tec::Server#` If you receive no output from the command it may mean that your T/EC server installation is not complete. The quickest solution may be to reinstall it. Note that this assumes that you are sending events to a T/EC server in the same TMR. If you are sending events to a connected TMR you will need to update the logfile configuration file, as described in 11.3.2.1, "The Adapter Configuration File" on page 362.
Is the adapter running?	Enter command `ps -ef \| grep logfile`. You should see a process executing program tecad_logfile. If not, you can start it using the following command: `/etc/Tivoli/tecad/bin/init.tecad_logfile start`
Does the event arrive in the T/EC server reception log?	Enter the command `wtdumprl -o DESC \| more`. You should see the event you generated at the top of the output. If you do not, re-check the preceding steps. If the event has arrived it should appear as follows: `1~ 11347~ 0~ 840555441(Aug 20 11:37:21 1996)` `### EVENT ###` `Su_Success;hostname=rs600020;...` `### END EVENT ###` `PROCESSED~' '` Instead of the final line saying PROCESSED, you may see a line like this: `PARSING_FAILED~ 'Line 1: Su_Success undefined'` This means that you do not have the event class defined in your rulebase. You need to import the baroc file, compile it, reload the rulebase and stop and restart the event server. This process is described in detail in 10.3.3, "Importing Event Classes" on page 332. For the logfile adapter, use the following sequence of commands: `wimprbclass tecad_logfile.baroc your_rule_base` `wcomprules your_rule_base` `wloadrb your_rule_base` `wstopesvr` `wstartesvr`

11.3.2 Configuration of the Logfile Adapter

After installation the logfile adapter should work with its default configuration. However, you will probably want to customize it for your local environment. We can think of customization in two parts:

1. Changing the way the adapter interacts with the event server
2. Adding other message types and message sources

We discuss the second, more complex customization process in 12.2, "Extending the Logfile Adapter to Manage a Distributed Application" on page 393. To change the way the adapter interacts with the event server you need to modify its configuration file /etc/Tivoli/tecad/tecad_logfile.conf. The file can either be modified directly or updated centrally using the Adapter Configuration Facility.

We first look at the file itself and then describe ACF. You should note that the format of the adapter configuration file is the same for all event adapter types.

11.3.2.1 The Adapter Configuration File

For full details of the entries in the file, refer to the adapter documentation. The following notes discuss the settings that you are most likely to want to alter.

1. Server location and secondary server

 The T/EC Logfile Adapter communicates with the event server using this address. You can define more than one T/EC server, separated with commas. If the first server is not available the adapter tries to communicate with the secondary server. There are three formats for the ServerLocation line:

 - ServerLocation=@EventServer

 There can only be one T/EC server within a TMR, so using this default entry causes the adapter to refer to the EventServer object and use TMP communications services to send events to it.

 - ServerLocation=@EventServer#rs600024-region

 In this case we are sending events to a T/EC server in a connected TMR. The remote TMR is called rs600024-region.

 - ServerLocation=rs600020

 This syntax for the server location should be used only for event adapters that are installed on systems without the Tivoli management platform installed. The adapter must be linked with different libraries so that it uses simple TCP sockets instead of the TMP services. The event adapter host must be able to resolve the host name (rs600020 in this case) into an IP address.

 It is possible to specify a list of alternate event servers here, by specifying a list of destinations separated by commas. For example we could have specified:

   ```
   ServerLocation=@EventServer,@EventServer#rs600024-region
   ```

 In this case the event adapter will try first to send an event to the T/EC server in the local TMR. If that fails it will send it to T/EC in the rs600024 region.

2. Event buffer

 If the logfile adapter cannot establish a connection to the event server, it can store the event in a flat file until the server is available again. The name of this file is specified as follows:

   ```
   BufEvtPath=/etc/Tivoli/tec/logfile.cache
   ```

3. Polling interval

 With this entry you can decrease the processor load of the logfile adapter. The polling interval is measured in seconds and determines the time between read operations on the file.

4. Filtering of events

 With filtering you can avoid unproductive network traffic and event server load. Before the logfile adapter sends any event to the event server it checks the event against the defined filters. You need to refer to the event class definition file tecad_logfile.baroc to decide which events interest you.

The entry for an event filter has the following syntax:

```
Filter:class_name;slot=value;slot=value;...
```

For example, the following entry would block any events that only match the base logfile class (that is, messages for which there is not a specific event class in the baroc file).

```
Filter:Class=Logfile_Base
```

11.3.2.2 Starting and Stopping the T/EC Logfile Adapter

You should always start the adapter with the command `init.tecad_logfile start`. The reason is that the logfile adapter receives syslog events by adding an entry to the syslogd daemon configuration file /etc/syslogd.conf. This entry causes the demon to write all system messages into a *named pipe*. For syslogd to be able to write to the pipe, it must be started after the logfile adapter has started to read from it. The `init.tecad_logfile` command guarantees that these operations are synchronized.

11.3.2.3 Updating Event Adapter Configurations Using ACF

The T/EC Adapter Configuration Facility (ACF) provides a GUI interface to configure the T/EC adapters. This eliminates the need to manually edit the files and allows you to update multiple adapters centrally. The adapters can be distributed using the Tivoli framework.

We installed the T/EC Adapter Configuration Facility on our TME server (node rs600024). The installation follows the standard TME product installation process. You will find information about installing and using this product in *Tivoli/Enterprise Console User Guide, Volume 1*.

First we added the adapter as a managed resource. From the Tivoli desktop do the following:

- Double-click on the icon for the policy region where you want to place the adapter configuration profiles.
- Select `Properties` followed by `Managed Resources` from the pull-down menu (see Figure 343 on page 364).

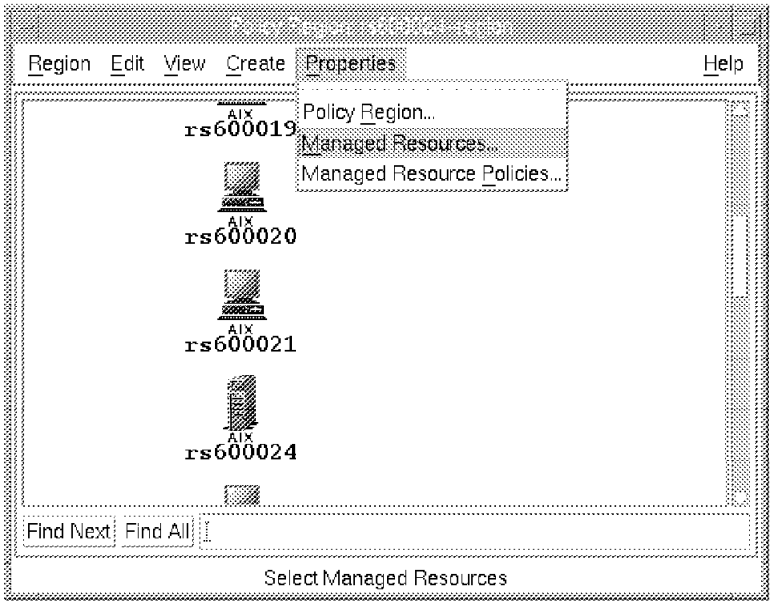


Figure 343. Editing Policy Region Managed Resource List

- Move the resource named ACP to the Current Resources list (see Figure 344).

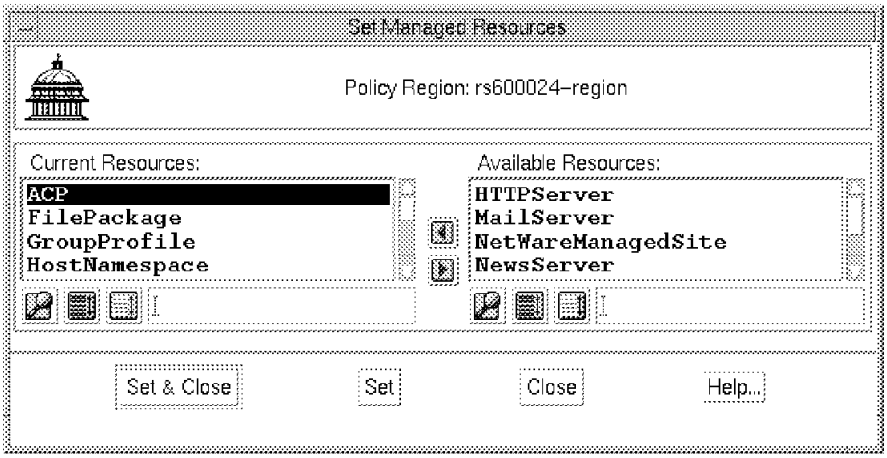


Figure 344. Adding Adapter Configuration Profiles to the Policy Region

- Select **Set & Close**.
- Assign ACF roles to the administrator who will update the adapter configurations (see Figure 345 on page 365).

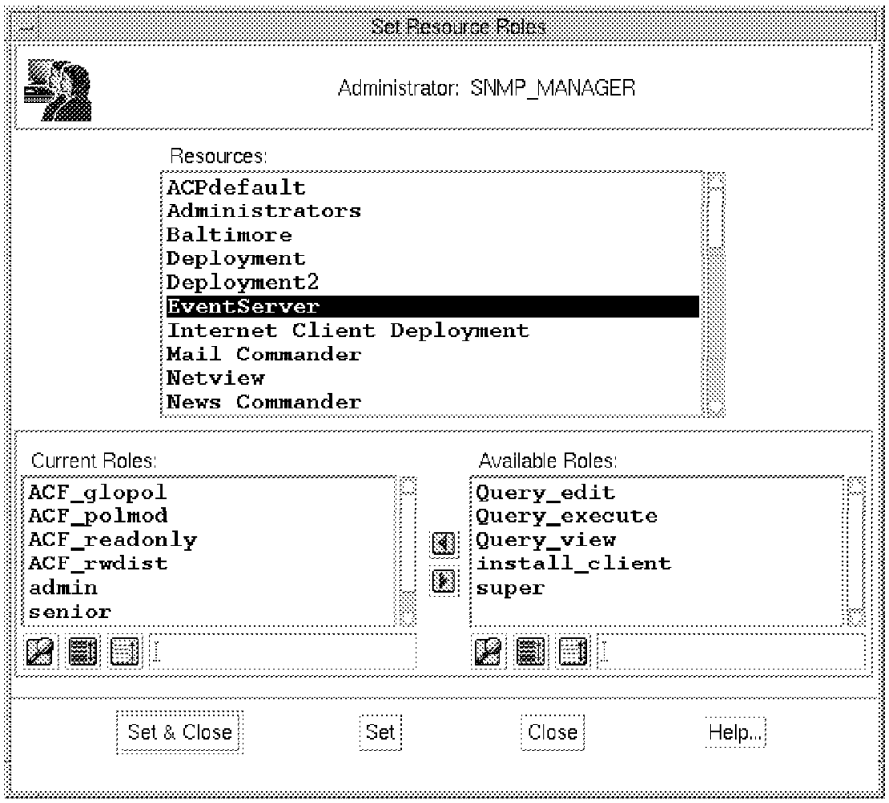


Figure 345. Assigning ACF Administration Roles

- Create a profile manager to contain the ACP profiles and then within the profile manager, create a profile. Figure 346 shows the dialog to create a profile called called Logfile_Adapter.

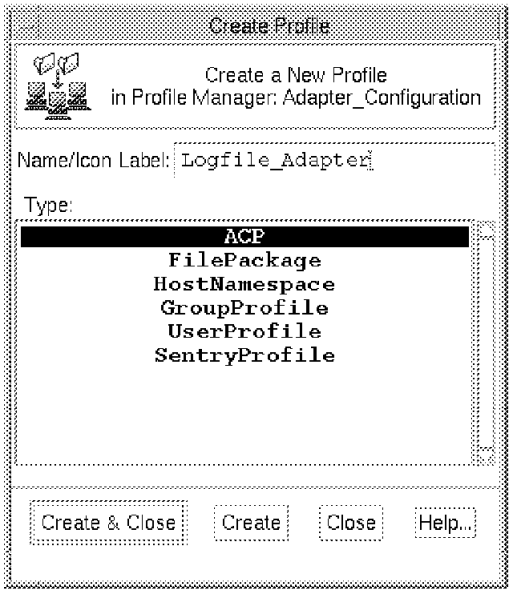


Figure 346. Create a New ACF Profile

Creating an ACP Profile from the Command Line: From the command line, enter:

```
wcrtprf "@ProfileManager:Adapter_Configuration" ACP Logfile_Adapter
```

- Using the right hand mouse button on the icon for your new profile, select **Edit Profile**, and then select **Edit** followed by **Add Entry** from the menu bar.
- You will be prompted to identify which type of adapter you want to define. Click on **tecad_logfile** followed by **Select & Close** (see Figure 347).

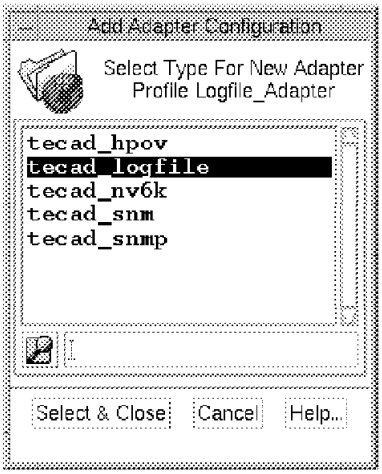


Figure 347. Selecting the Adapter Type for an ACF Profile Entry

Next you will be presented with a dialog in which you can modify the configuration for the logfile adapter. This helps avoid errors that may occur if you edit the file directly. Figure 348 shows a typical panel from the dialog.

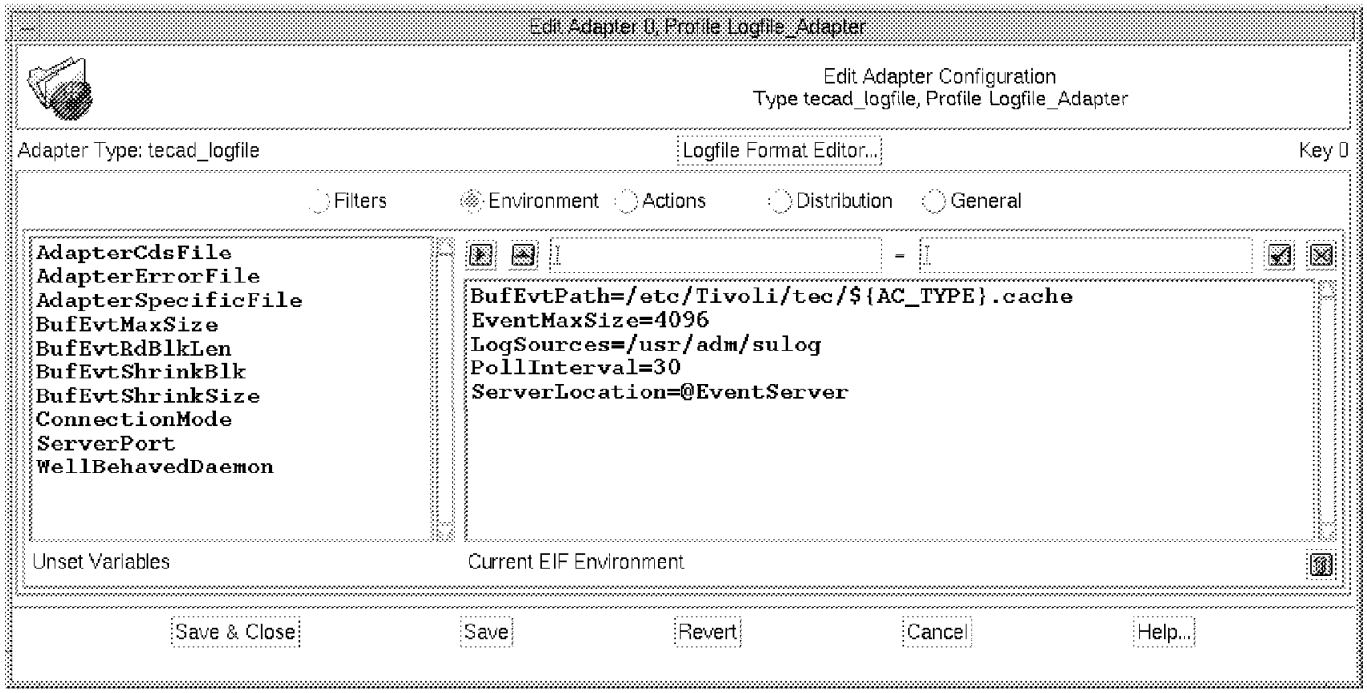


Figure 348. Updating the Logfile Adapter Configuration

In addition to updating the tecad_logfile.conf definitions, ACF also distributes other files that the adapter needs. All of the available configuration files are held in the directory /usr/local/Tivoli/bin/generic_unix/TME/ACF_REP.

Figure 349 on page 367 shows the dialog for modifying the destination of these additional files. In this case we had to update the destination for the tecad_logfile.fmt file, because the TMR server was a different machine architecture to the target.

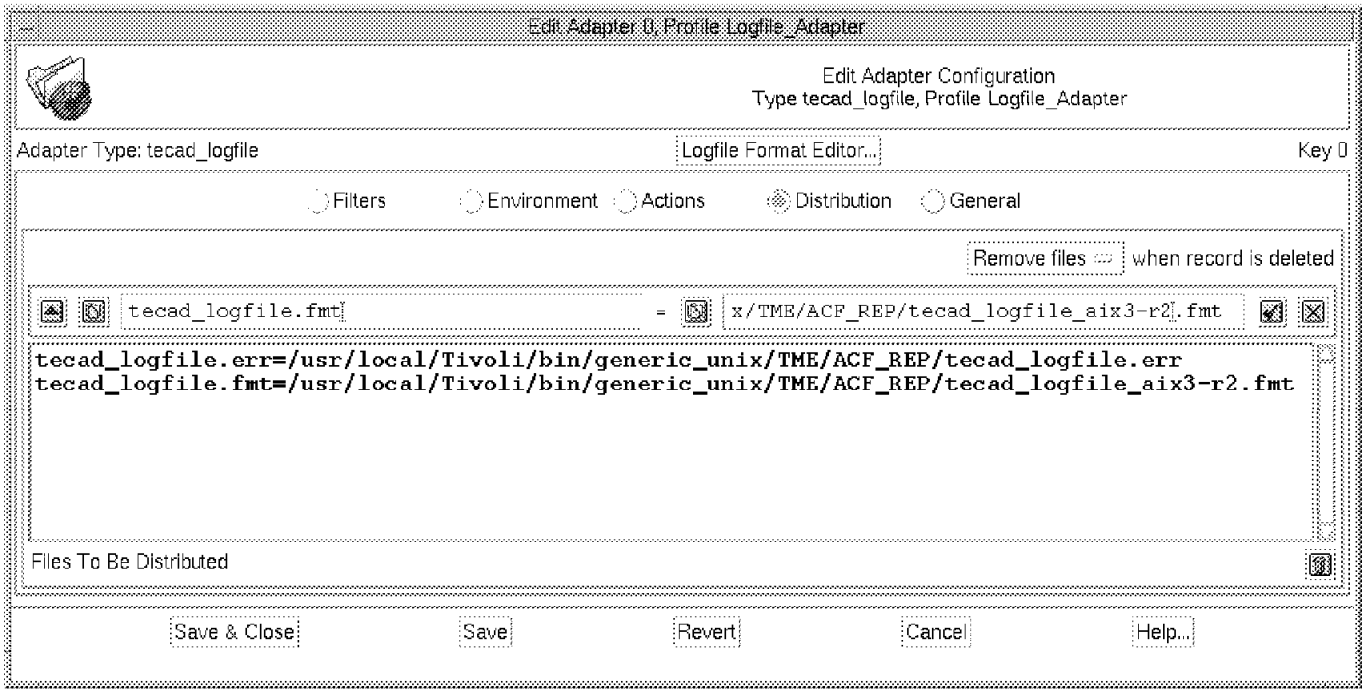


Figure 349. Defining the Destination for Additional Adapter Files

To install the updated configuration, add the nodes that have the logfile adapter installed as subscribers and distribute the profile in the normal way. The logfile adapter must already be installed on the node that is a subscriber. Otherwise, you will not be able to perform the subscription.

You can check if the logfile adapter is configured on a particular machine by issuing the `wlsinst` command. Look for the following entry:

```
Tivoli/Enterprise Console Logfile Adapter
```

11.4 NetView for AIX and Openview Adapters

SNMP network management stations provide an important source of events, allowing information from devices outside the normal range of the TME platform to appear in the T/EC. There are three event adapters for the three network management systems (HP Openview, SunNet Manager and NetView for AIX) included in the T/EC. The events generated by these adapters are not *only* SNMP traps. They also generate internal events to reflect status changes of nodes in the network.

This section covers the NetView for AIX adapter. This adapter is virtually identical to the Openview adapter with the exception of the naming policy within the configuration files. However, the examples shown will apply to both. In fact, there is a new, improved adapter for NetView for AIX, which we call the NetView Ruleset Adapter. It is covered in 11.5, “The NetView for AIX Ruleset Event Adapter” on page 370. The main difference between the NetView ruleset

adapter and the adapter described here is that it uses NetView for AIX rulesets for event filtering, which means that you need AIX Version 4 or higher to use it.

11.4.1 Installation and Initial Configuration

You install the NetView for AIX or Openview adapter using the standard TME product install process (as described in 5.2, “Installation” on page 109). When the adapter has been installed, the `wlsinst` command will identify it as:

```
Tivoli Enterprise Console NetView/6000 adapter
```

You should refer to the reference guide for this adapter, *Tivoli/Enterprise Console Event Adapter: IBM NetView*, before proceeding.

11.4.1.1 Configuration: Adapter Configuration

The adapter is a daemon that runs as an extension to NetView for AIX. Therefore, you start it like any other NetView daemon, by entering `ovstart tecad_nv6k` or using the Smit option.

If the adapter fails to start and the return code is 1, the reason is that it is unable to locate the T/EC server. The configuration file is identical to the file used by the logfile adapter, which we described in 11.3.2.1, “The Adapter Configuration File” on page 362. In our case we needed to tell the adapter to send events to a T/EC server in a different TMR, so we updated the server location entry in tecad_nv6k.conf as follows:

```
@EventServer=root_region#rs600024
```

Before attempting any further configuration we copied all of the adapter files into an editing area. The configuration files for tecad_nv6k are as follows:

tecad_ov.baroc Event class definitions

tecad_snmp.baroc Event class definitions

tecad_nv6k.baroc Event class definitions

tecad_nv6k.cds NetView event to baroc mapping rules

tecad_nv6k.conf Adapter configuration file

tecad_nv6k.oid SNMP object ID to text mapping file

By default, the definitions in these files will convert many of the standard NetView for AIX events through to T/EC. However, SNMP traps from other device types will not be converted unless you customize the definitions. We show an example of this in 12.3, “Extending the NetView for AIX/OpenView Adapters” on page 413.

11.4.1.2 Customization: NetView Adapter Event Classes

To prepare the T/EC server to receive the events you need to import the event class definitions. According to the adapter reference guide, the installation script will install the adapter into the Default rulebase. We recommend that you use your own rulebase, so you will have to either copy the classes from the Default rulebase (as described in 10.3.2, “Initial Configuration of the T/EC Rulebase” on page 329) or import them manually. There are three baroc files containing the event classes used by this adapter: tecad_snmp.baroc, tecad_ov.baroc and tecad_nv6k.baroc. Of these, tecad_nv6k contains entries that are subclasses of classes within tecad_ov. This means that the order in which you import the files is important. The following sequence of commands will

import the event classes, compile the rulebase, load it and restart the event server (the rulebase name is NetView):

```
wimprbclass /etc/Tivoli/tecad/etc/tecad_ov.baroc NetView
wimprbclass /etc/Tivoli/tecad/etc/tecad_nv6k.baroc NetView
wimprbclass /etc/Tivoli/tecad/etc/tecad_snmp.baroc NetView
wcomprules NetView
wloadrb -u NetView
wstopesvr
wstartesvr
```

You can also perform the same sequence of actions from the GUI, as shown in 10.3.2, “Initial Configuration of the T/EC Rulebase” on page 329.

11.4.1.3 Customization: Defining Sources and Event Groups

On the output side of the event server, you need to create event source and group definitions so that the NetView events can be assigned to a console. The source name for the adapter is NV6K. The following sequence of commands adds the source, creates two groups (one with a filter that passes all NetView events and another that passes only Node Down events), and finally assigns the event groups to administrator Paul′s event console:

```
wcrtsrc -l "NetView/6000" NV6K
wcrteg -b ibm.xpm NETVIEW_EVENTS
waddegflt -s NV6K NETVIEW_EVENTS
wcrteg -b ibm1.xpm NV_NODE_DOWN
waddegflt -c OV_Node_Down -s NV6K NV_NODE_DOWN
wassigneg @Paul NETVIEW_EVENTS admin user
wassigneg @Paul NV_NODE_DOWN admin user
```

Notice that we can easily create the Node Down filter because each different type of event from the NetView for AIX adapter is defined as a separate event class. The event group window resulting from these definitions is shown in Figure 350 and the event display for the second group is shown in Figure 351 on page 370.

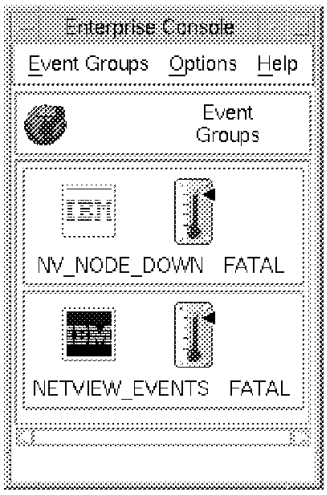


Figure 350. Available Event Groups

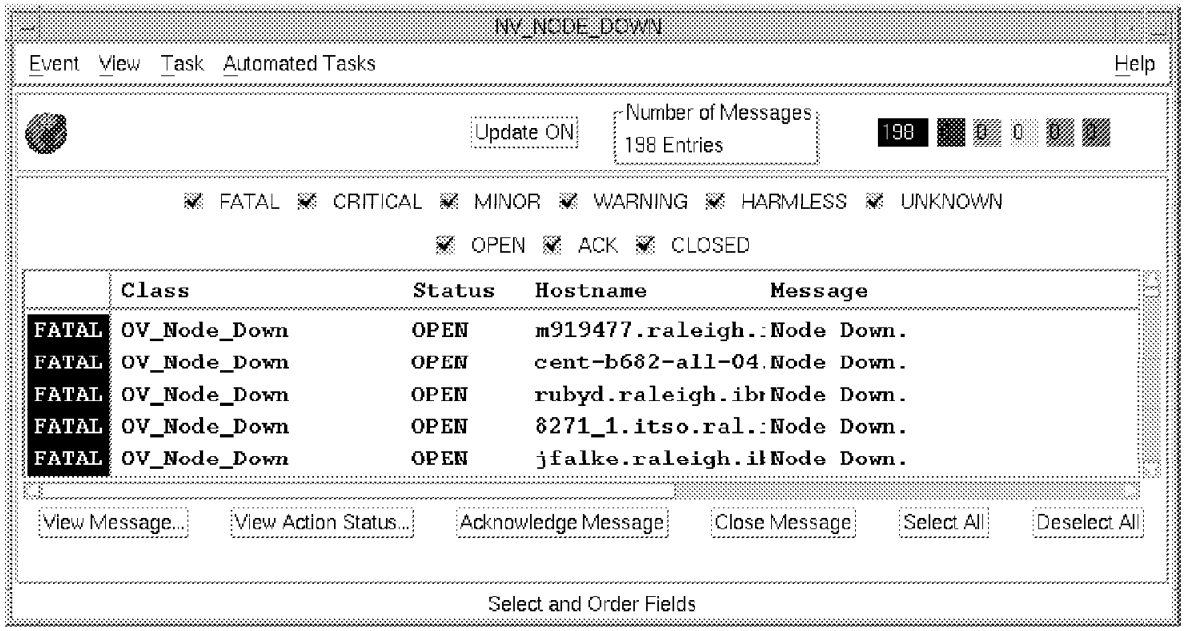


Figure 351. Events with Source NV6K and Class OV_Node_Down

11.5 The NetView for AIX Ruleset Event Adapter

The NetView for AIX event adapter described above relies on configuration files to define which events to send to the T/EC and the details of the conversion. The NetView for AIX ruleset adapter uses a NetView event ruleset to decide which events to pass to the T/EC. Furthermore, the event conversion is a generic process, so there is no need to have specific mapping definitions for each event type.

Figure 352 on page 371 illustrates the different paths taken by events in the two adapters. The older adapter uses the OvSNMP API to receive raw trap information from the NetView trapd daemon and then generates T/EC events directly, based on the configuration files. The ruleset adapter code is embedded in the nvserverd daemon (which is also the daemon that passes events to NetView event displays). It registers a NetView ruleset to be executed by nvcorrd, the rule processing daemon. Any event that is forwarded by the ruleset is then converted into a T/EC event. You can think of the ruleset as a *smart filter*: using the power of the NetView rules to prevent the T/EC server from becoming overloaded.

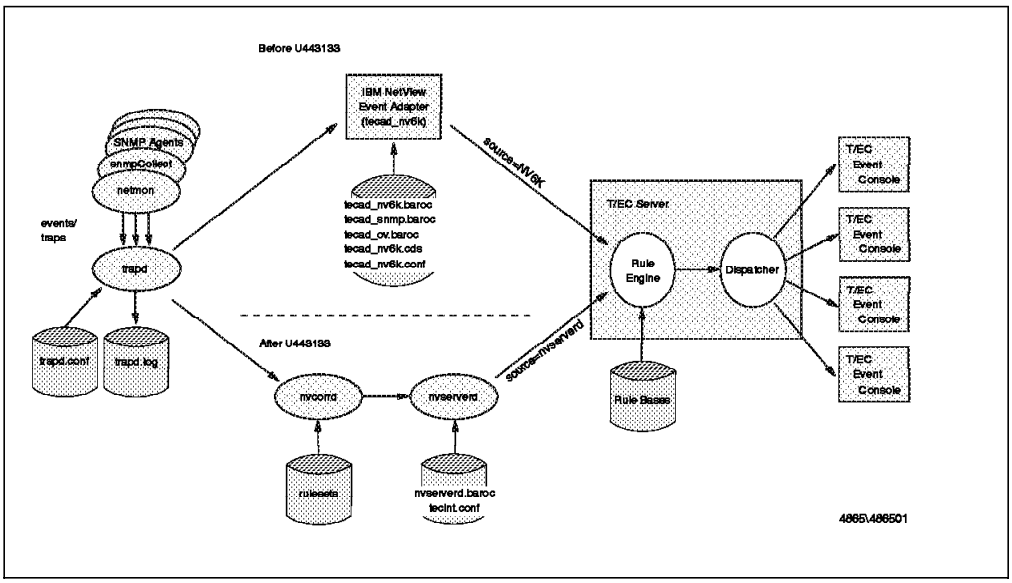


Figure 352. Comparing the Two NetView Event Adapters

11.5.1 Installation and Initial Configuration

Unlike the other event adapters, which are provided as additional TME products on the T/EC CD, the NetView ruleset adapter is built in to NetView. The installation process is therefore a question of applying PTF U443133 using the standard AIX Installp function and then configuring the adapter.

11.5.1.1 Configuring the Ruleset Adapter

Configure the adapter as follows:

1. On the NetView machine, enter `smit nv6k`.
2. Select **Configure** and then **Configure for Tivoli**.
3. Define the host name of the event server and the NetView ruleset that you want to use to control event forwarding. Figure 353 shows the values we entered in this dialog.
4. Select **OK** and then **Exit**.

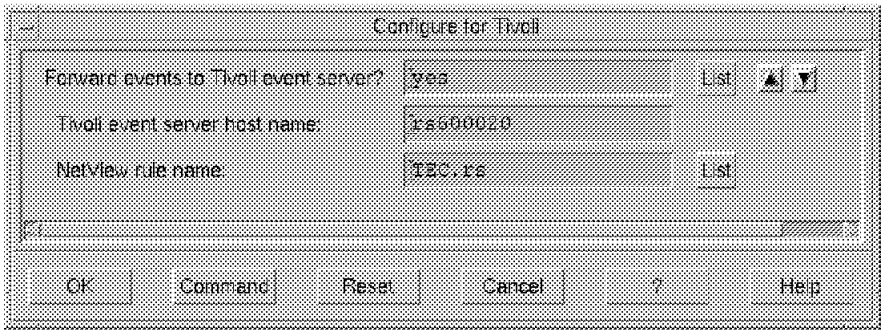


Figure 353. Configuring the NetView Adapter Using Smit

The values shown in Figure 353 instruct the nvserverd daemon to register ruleset TEC.rs for T/EC event forwarding and to send events to the T/EC server on host rs600020. The result of this configuration dialog is to update the /usr/OV/conf/tecint.conf file. The file will contain two lines, as follows:

```
ServerLocation=rs600020
TecRuleName=TEC.rs
```

Note that the server location is an IP node name, not the @EventServer#TMR-region format. This indicates that the ruleset adapter does not use TME platform services to deliver events.

To cause these changes to take effect, stop and restart the nvserverd daemon by using the following commands:

```
ovstop nvserverd
ovstart nvserverd
```

11.5.1.2 Customization: NetView Ruleset Adapter Event Class

To prepare the T/EC server to receive the events, you need to import the event class definitions. There is only one baroc file containing the single event class used by this adapter: /usr/OV/conf/nvserverd.baroc.

The contents of the nvserverd.baroc file are shown in Figure 354

```
#!/bin/ksh
#
# Licensed Program Product:
#   NetView for AIX  V4R1
#
# (C) COPYRIGHT International Business Machines Corp. 1992,1994
#     All Rights Reserved
# US Government Users Restricted Rights - Use, duplication,
# or disclosure restricted by GSA ADP Schedule Contract with
# IBM Corp. and its licensors.
#

################################################################################
#                                                              #
#  NetView generic trap                                             #
#                                                              #
################################################################################
TEC_CLASS :
      Nvserverd_Event ISA EVENT
      DEFINES {
            source: default = nvserverd;
            nv_enterprise: STRING;
            nv_generic: INT32;
            nv_specific: INT32;
            nv_var1: STRING;
            nv_var2: STRING;
            nv_var3: STRING;
            nv_var4: STRING;
            nv_var5: STRING;
            nv_var6: STRING;
            nv_var7: STRING;
            nv_var8: STRING;
            nv_var9: STRING;
            nv_var10: STRING;
            nv_var11: STRING;
            nv_var12: STRING;
            nv_var13: STRING;
            nv_var14: STRING;
            nv_var15: STRING;
            };
END
```

Figure 354. The nvserverd.baroc File

The slots defined in the class have the following meanings:

source	Set to nvserverd.
nv_enterprise	The enterprise ID (MIB object) from the trap. (This is the NetView enterprise ID for internally generated events.)
nv_generic	The generic trap number.
nv_specific.	The specific trap number.
nv_var*n*	The values of the variables within the trap or event. Fifteen variables are defined in the baroc file. Most traps only have a few variables, so limiting it to 15 is not generally a problem.

The adapter also sets the event severity and the event message text (*severity* and *msg* are slots inherited from the base event class, so they are not redefined in the nvserverd.baroc file). These values are translated from the values defined when the event is configured in NetView. You perform the configuration by selecting **Options**, **Event Configuration** and then **Trap Customization** from the NetView menu bar.

You need to import the event class into T/EC, compile the rulebase, load it and restart the event server. Look at 10.3.2, “Initial Configuration of the T/EC Rulebase” on page 329 for details. From the command line the sequence of commands is:

```
wimprbclass /usr/OV/conf/nvserverd.baroc NetView
wcomprules NetView
wloadrb -u NetView
wstopesvr
wstartesvr
```

11.5.1.3 Customization: Defining Sources and Event Groups

On the output side of the event server, you need to create event source and group definitions so that the events from the ruleset adapter can be assigned to a console. The source name for the adapter is *nvserverd*. You can find an example of creating the necessary source and group definitions in 10.3.4, “Defining T/EC Groups and Sources” on page 336.

11.5.2 Configuring the NetView Rulesets

The original NetView for AIX event adapter is pre-configured to pass specific events on to the T/EC. In the case of the NetView Ruleset dadpter you have to create a ruleset to define which events will be passed. Although this is an extra step, it is not complex because the NetView ruleset editor is very simple to use.

Only one ruleset can be defined to send events to the T/EC server at any one time. The process below shows how we created a small NetView for AIX ruleset that has only two events included for forwarding to the T/EC server. If you want to understand in detail how to use the NetView ruleset facility, refer to *Examples of Using NetView for AIX Version 4*, SG24-4515.

The completed ruleset is shown in Figure 355 on page 374. You can see that it contains two Trap Settings decision nodes. If the event matches either of these nodes, it is passed on to the Forward node. This causes the event to be passed on to nvserverd, which then generates a T/EC event from it. There are three ways in which an event can pass the ruleset and therefore be sent to T/EC:

1. If it encounters a Forward action node (as in this case)

2. If it encounters an Override action node
3. If the default behavior for the ruleset is to forward events and the event does not encounter a Block action node

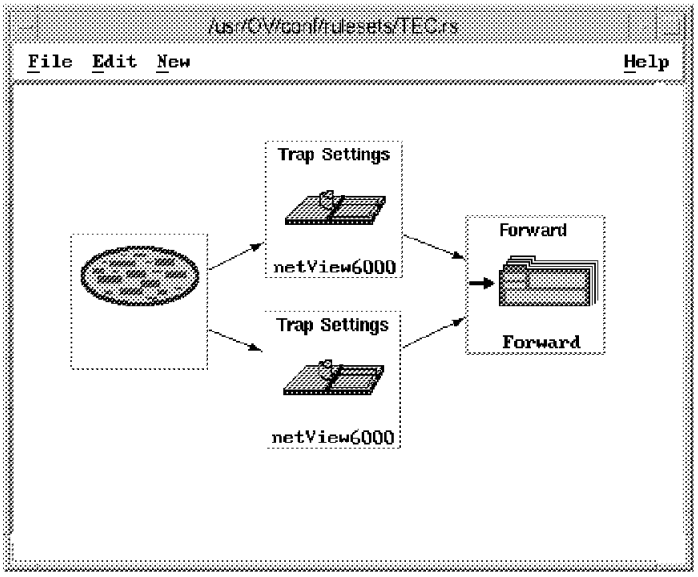


Figure 355. The TEC.rs Ruleset Definition

The Trap Settings nodes in this case are set to pass Node Up and Node Down events. Figure 356 on page 375 shows how the Trap Settings node for Node Down was configured.

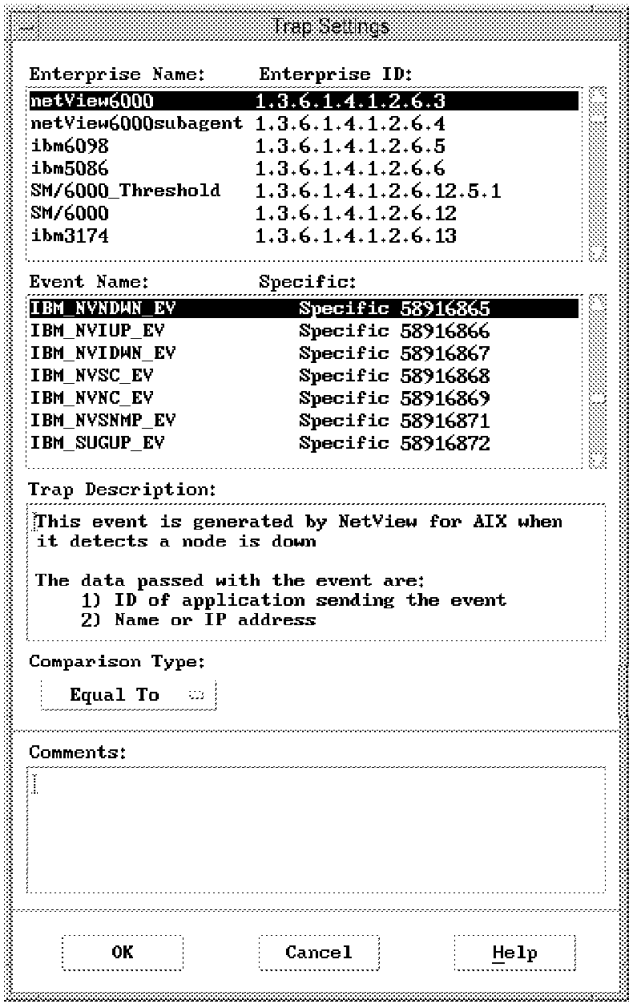


Figure 356. The Node Down Definition from TEC.rs

Finally you can test the configuration by creating a NetView event from the command line. Enter the following command on the NetView machine to create a Node Down event:

```
event -e NDWN_EV
```

You should see the Node Down event appear in the console display. If you do not, follow the problem determination procedure described in Table 13 on page 360.

11.6 The SNMP Adapter

We did not explore the SNMP adapter in any detail. The most effective method to send SNMP events to the T/EC server is by using one of the network management station adapters. You would only consider using the SNMP adapter if you did not have a network management station installed.

11.7 The Tivoli Sentry Event Adapter

Sentry can generate events and send them to T/EC, so technically it is an event adapter. It is unusual in that the function is buried within the application, rather than being a separate program as are most other event adapters.

11.7.1 Importing Sentry Event Classes

As with any event adapter, before you send events from Sentry you need to import the event classes into the active T/EC rulebase. We described this process in 10.3.3, “Importing Event Classes” on page 332. When you install Sentry it places a great many baroc files that define its event classes. These files are, by default, located in directory /usr/local/Tivoli/bin/generic/SentryMonitors. The class structure assigns a different event class to each Sentry monitor event. Each of these classes is a subclass of the general Sentry event class named *Sentry2_0_Base*, which is defined in the Sentry.baroc file.

The result of this is that for any given Sentry monitor type you need to import at least two baroc files. Furthermore, it is important to import them in the right order, to make sure that the class hierarchy can be resolved. For example, to use all of the UNIX monitors it is necessary to import the following sequence of class definitions:

```
Sentry.baroc
tivoli.baroc
universal.baroc
```

11.7.2 A Sentry Monitor Example

To illustrate how T/EC events are generated from a Sentry monitor we created a simple example. In this example we are monitoring whether the NetView for AIX *netmon* daemon process is active or not. We showed how to set up Sentry monitors using the GUI in Chapter 9, “TME 10 Distributed Monitoring” on page 293, so for this example we chose to use the command line interface.

The following commands create the monitor and install it:

```
wcrtprfmgr @rs600024-region NetView_Monitor
wln @ProfileManager:NetView_Monitor Operations
wcrtprf @ProfileManager:NetView_Monitor SentryProfile netmon_daemon
wsub @NetView_Monitor @ManagedNode:rs600021
waddmon UNIX_Sentry "daemon" -a "netmon"  -t "5 minutes" -c critical \
  -i -R "==" down -T CRITICAL EventServer netmon_daemon
wdistrib -l over_all  @SentryProfile:netmon_daemon @ManagedNode:rs600021
```

If you want to know more about the CLI commands, read the man pages for them. We simply describe what each of the above commands does, without describing the options in detail:

wcrtprfmgr This creates a new profile manager named NetView_Monitor.

wln This places the new profile manager within the policy region called Operations on the desktop of the current administrator.

wcrtprf This creates a Sentry profile called netmon_daemon in the profile manager.

wsub This subscribes the system running NetView for AIX (rs600021) to the profile manager.

waddmon This defines a monitor entry within the profile, checking the status of netmon every five minutes and sending a critical T/EC event if it is down.

wdistrib This distributes the new profile to rs600021.

These commands result in the objects and windows shown in the NetView_Monitor profile manager, the netmon_daemon profile and the profile entry within it appearing on the desktop, as shown in Figure 357.

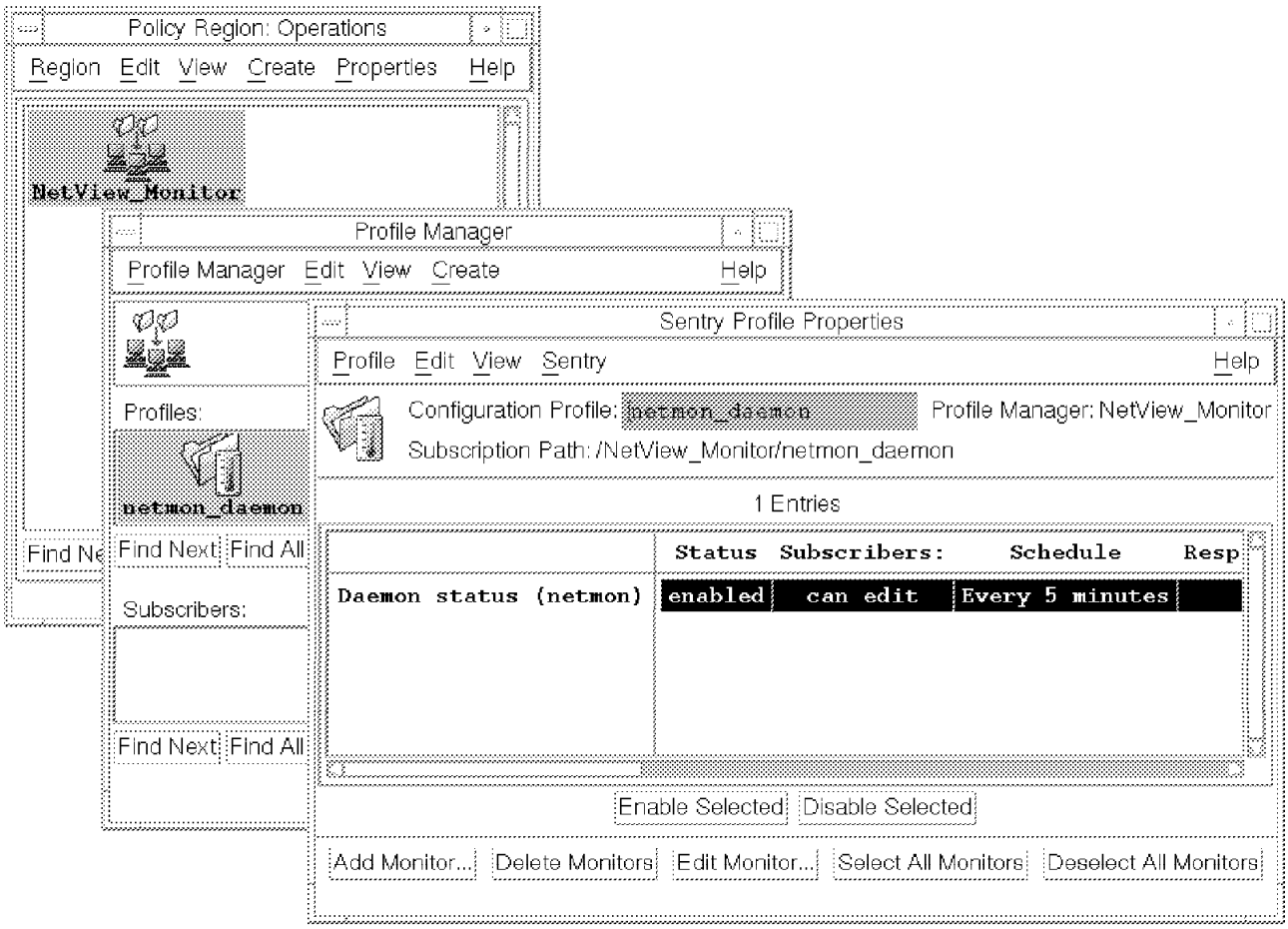


Figure 357. Sentry Monitor Components

If you open the profile entry, you will see the display shown in Figure 358 on page 378. Notice the Enterprise Console specification toward the bottom of the window.

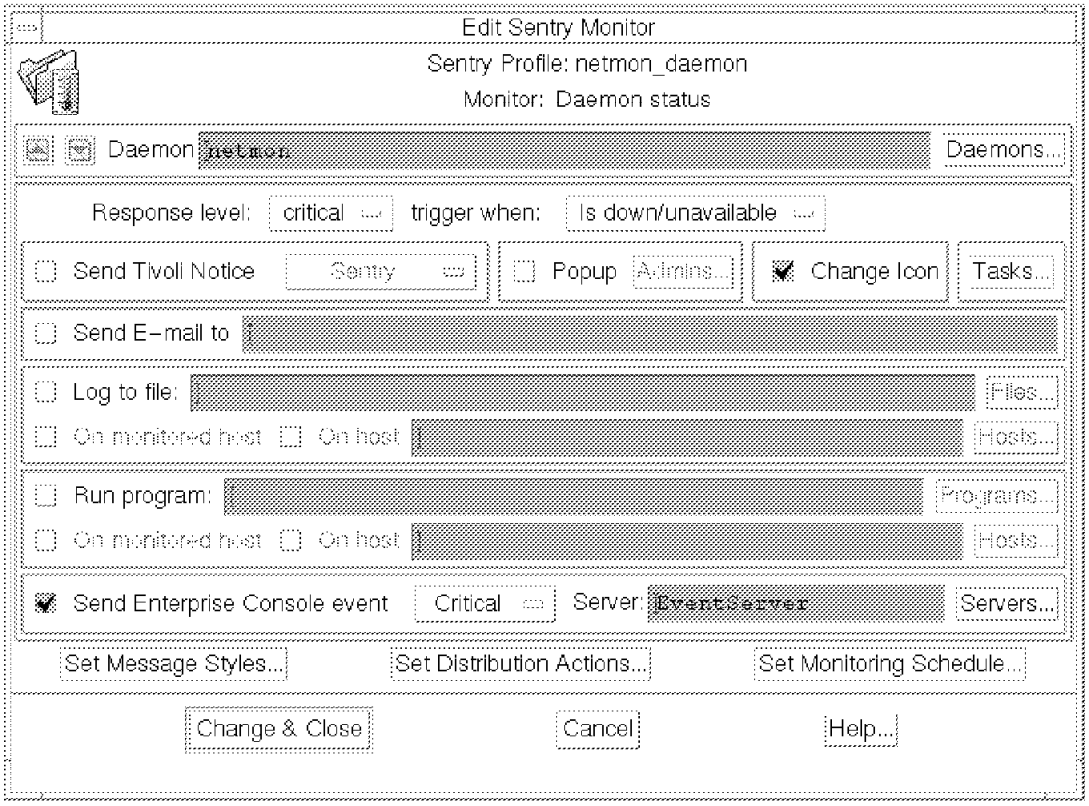


Figure 358. Monitor for netmon Status with T/EC Event Forwarding

When we used the waddmon command to define the monitor we specified the -i flag, meaning that an icon will change status when the threshold is triggered. Therefore to complete the setup we also had to add a Sentry monitoring collection and associate the monitor with it. See 9.2.7, “Creating an Indicator Collection” on page 310 for details on how to do this.

To trigger an event we simply stopped the netmon daemon on the monitored (rs600021) node and waited. After a while the indicator on the monitoring icon moved to critical and the wtdumprl command showed that an event had arrived at the T/EC.

In order to see this event in a console display we needed to add an event source for the Sentry events and define a filter that would pass them to the console. Refer to 10.3.4, “Defining T/EC Groups and Sources” on page 336 for details of how this is done. The resulting events as seen in the T/EC console display is shown in Figure 359 on page 379.

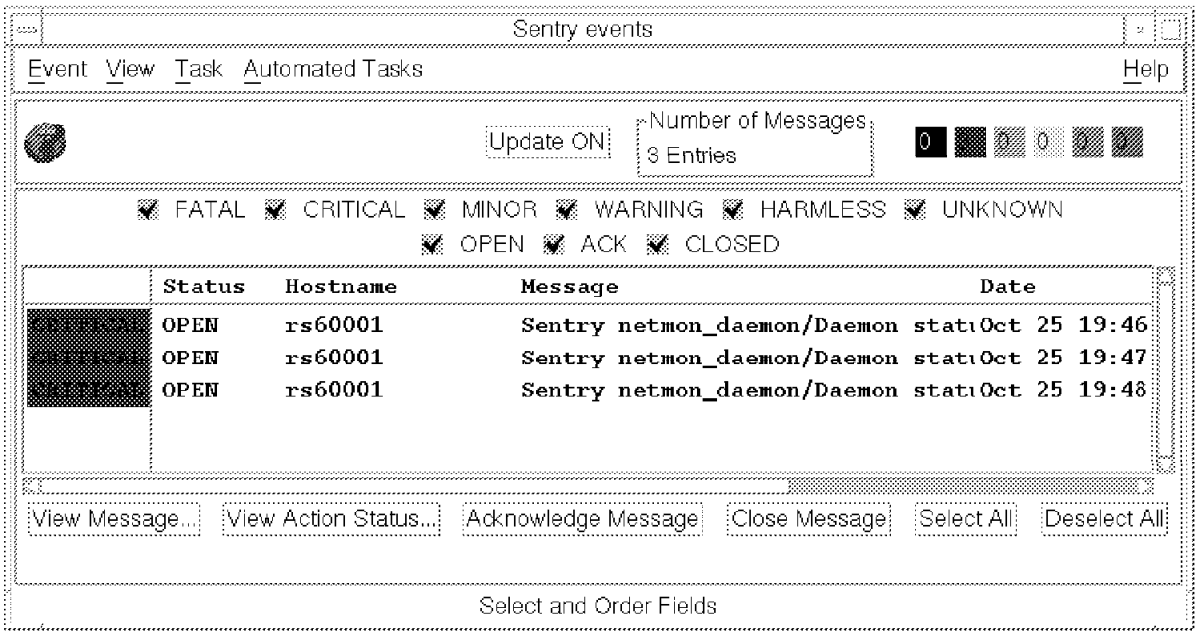


Figure 359. Sentry Events in the Console Display

11.8 Windows NT Adapter

The NT adapter is a variation of the Logfile adapter for UNIX nodes. It is designed to convert errors recorded on Windows NT servers into T/EC events. An alternative way to do this is to use NT Sentry monitors. The NT Logfile adapter is less flexible than defining a Sentry monitor, but it requires less initial setup. In addition, it will operate on any NT node, not only on an NT Managed Node.

The Windows NT adapter runs as an NT service. The adapter will send a T/EC event from information contained in the NT system, application and security logs. These logs are updated by the NT operating system, and by applications running on the system. To review which events the adapter forwards to the T/EC, look at file tecad_nt.fmt. This file contains information about how the log message is mapped into a baroc format and is in the same format as the UNIX logfile adapter .fmt file. See 12.2, “Extending the Logfile Adapter to Manage a Distributed Application” on page 393 for an example on how to modify this file so that the adapter will forward additional log messages.

The file tecad_nt.con is the equivalent of the .conf file that we have seen in the case of all of the other adapters (except for Sentry). Refer to 11.2, “Adapter Configuration Files” on page 356 for a description of the fields in this file.

11.8.1 Installing the NT Adapter

Note that the following installation instructions apply to the version of the NT adapter available at the time of writing. The installation process will be modified to be a standard TME product install in the near future. The NT logfile adapter is shipped on the T/EC CDRom under directory /CDROM/ADAPTERS/NT. It is a self-extracting executable file named TECADNTZ.EXE. To install the adapter:

1. Copy TECADNTX.EXE into a temporary directory on the NT system and execute it to unpack the files.

2. Edit the tecad_nt.con file to reflect the IP host name of your event server. Note that the version of the adapter that we used only supported TCP/IP socket communications with the event server, instead of using TME services. This means that you cannot use symbolic definitions such as @EventServer in the ServerLocation field.

3. Install the adapter itself by executing the command `tecinstl`, and specifying the disk letter and path of the directory where you want the adapter code to be placed. In addition to installing the adapter files, the command will also update the NT registry and start the adapter as a background service.

11.8.2 Configuring T/EC for the NT Logfile Adapter

The NT logfile adapter generates only one event class, namely NT_Base. The events also all have the same source slot value of NT. They are further identified by the sub_source event slot whose value is one of Security, Admin or System, depending on the event log from which it came.

The event class defininition is provided in the baroc file tecadnt.brc, which is installed as part of the NT adapter. To enable the T/EC to display NT events you use the following sequence of actions:

1. Import the event class defined in the baroc file into the T/EC rulebase.
2. Compile the rulebase and stop and restart the event server.
3. Define an event source of NT and an event group filter that passes NT source events to a console.

See Chapter 10, “Introduction to the TME 10 Enterprise Console” on page 319 for a detailed description of these steps.

11.8.3 An Example of NT Logfile Adapter Events

We used the NT service status messages, recorded in the Security log to demonstrate the operation of the adapter. You can do this by selecting **Services** from the System Setup window in NT, selecting a service and clicking on **Stop**. The result is that a message is generated in the Security log and the NT logfile adapter passes it on to the T/EC. Figure 360 shows the message as it appears in the output of the wtdumprl command and Figure 361 on page 381 shows the event console display.

```
### EVENT ###
NT_Base;hostname=wtrO5119;origin=9.24.1O4.213;category=5;EventType=AuditS
Success;SID=WTRO5119\Administrator;sub_source=Security;
ID=593;msg='A process has exited:
 Process ID: 4289728544 User Name: Administrator Domain:
WTRO5119 Logon ID: (0x0,0xBC9C)'
END
```

Figure 360. wtdumprl Output for an NT Logfile Event

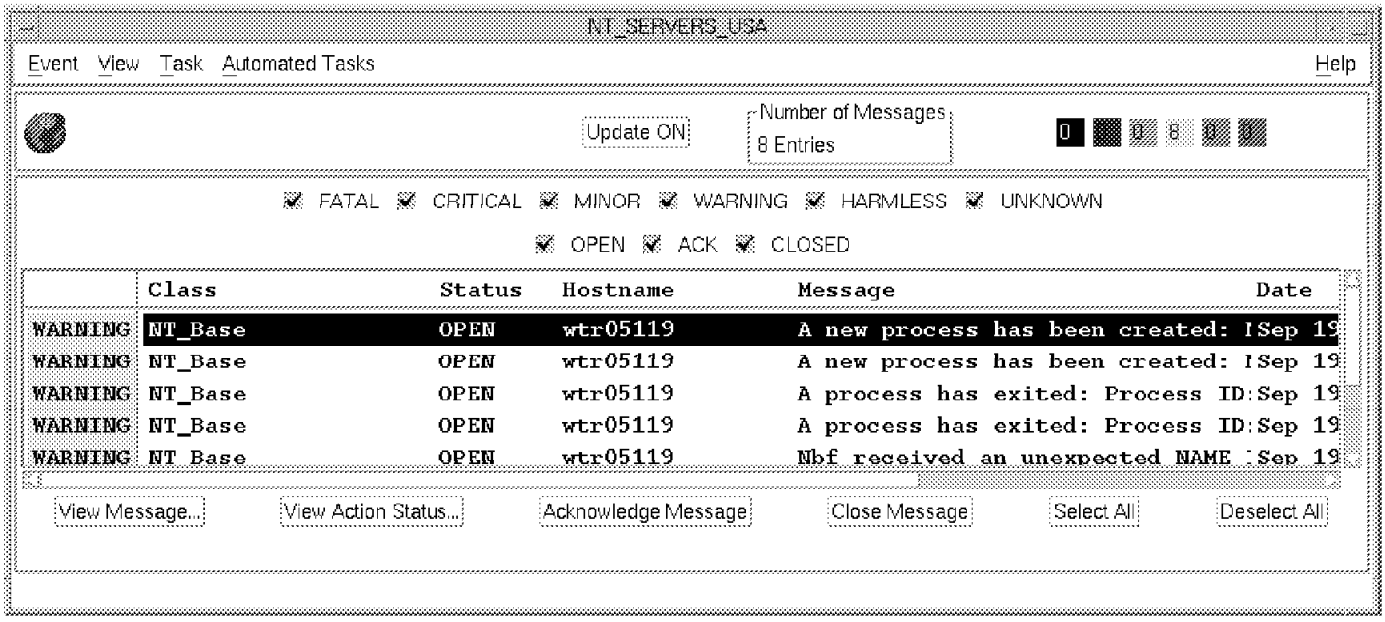


Figure 361. NT Events from the NT Adapter

11.9 Generating T/EC Events from the Command Line

The adapters we have described so far are designed to take existing event messages, in a variety of forms, and convert them into a baroc format. Usually, however, you will want to generate other events that the standard adapters do not handle. For example, you may have an application from which you want to pass error information to a central administrator. One approach is to extend the standard adapters to handle different messages. We show two examples of this in the next chapter: adding additional messages to the logfile adapter and additional SNMP traps to the NetView for AIX/Openview adapters.

Another way to get other event information into T/EC is to generate events using a command line function, either manually or within a program or shell script. There are two important commands that allow you to generate any T/EC event in this way. They are as follows:

postemsg This allows you to generate an event of any class and provide the slot values from the command line. The event is sent using normal TCP/IP sockets.

wpostemsg This is the same as postemsg, except that the event is sent using the Tivoli Management Platform remote procedure call function.

You can think of postemsg and wpostemsg as general purpose event adapters, for cases where the more specific adapters do not apply. They are also useful for testing your T/EC configuration in cases where the real event is difficult to re-create. There is an example of this in Table 13 on page 360.

Chapter 12. More Advanced TME 10 Enterprise Console Customization

In Chapter 10, “Introduction to the TME 10 Enterprise Console” on page 319 we described how the T/EC operates, and the steps required to install and configure it. In Chapter 11, “Tivoli/Enterprise Console Adapters” on page 355 we explored the capabilities of most of the available T/EC event adapters. It would be possible to install and run the T/EC using only the adapters as they are shipped. But for the T/EC to deliver the power of which it is capable, you will have to perform some further customization. In fact, many customers choose to employ Tivoli Professional Services to assist with this work.

In this chapter we look at some typical examples of the kind of customization that is possible. The chapter is broken into several sections:

- 12.1, “Creating Enterprise Console Rulesets” is an introduction to creating event rulesets illustrated with some simple examples.
- 12.2, “Extending the Logfile Adapter to Manage a Distributed Application” on page 393 uses a practical application example (MQ Series) to demonstrate how to make the logfile adapter deliver application messages to the T/EC and how to use the T/EC rules to automate the recovery process.
- 12.3, “Extending the NetView for AIX/OpenView Adapters” on page 413 shows how to configure the NetView for AIX and Openview adapters to forward SNMP traps from other sources to the T/EC.

12.1 Creating Enterprise Console Rulesets

The T/EC can receive events from many sources. Potentially there could be many events arriving each minute. Although the software is designed to cope with an average event arrival rate of 5 to 10 events per second (and bursts of even higher activity) an administrator cannot hope to assimilate events that fast. The purpose of the T/EC rules is to provide a mechanism to reduce the load on the administrator, by applying logic to the events as they arrive. Some of the things that event rule sets can be used for include:

- Event correlation. That is, analyzing event contents to try to eliminate events that are only *effects* but retaining events that are *causes*.
- Modifying event severity, to highlight particularly important problems.
- Running tasks against events to attempt to perform automated recovery from a problem.
- Removing or closing duplicated events, so that the administrator sees only one problem, instead of the same problem recurring many times.

We do not have room in this book to create examples that illustrate the many functions of the rules, but we give examples that show some of the more simple capabilities.

© Copyright IBM Corp. 1996

12.1.1 An Introduction to T/EC Rules

Although they vary enormously in complexity, every T/EC rule is comprised of two basic sections:

1. The *event* section which as a minimum identifes the event class that the rule will handle. It may also contain *conditions* defining criteria which the data in the baroc slots must meet.
2. The *action* section which consists of a list of actions to be taken if the event class and the other conditions are met. The actions may invoke function calls, known as *templates* which perform some action or diagnostic function.

The actions are conditional. That is, if one action fails none of the actions following it will be executed. Conversely, some actions can return multiple *true* results, in which case the actions following them will be executed multiple times. This function is most commonly used for correlation rules that search for matching events that have been previously received.

Most rules are *reception rules*, triggered by the arrival of a new event. However, they can also be triggered by an administrator action (closing or acknowledging and event, for example) and by a timer.

There are two ways to create T/EC rulesets, by using the GUI rule builder or by writing the rule code directly in the Prolog programming language. The rule builder is the easiest approach, but it does not permit you to generate some of the more complex rule constructs. We show one simple example of each method.

12.1.2 Using the Rule Builder

The T/EC rule builder is a GUI interface that allows the development and customization of simple T/EC rules. You should refer to the reference manual *Tivoli/Enterprise Console Rule Developer's Manual* for rule development when doing any ruleset work.

The example we show here is driven by the arrival of an event of class SM6K_ARM. The rule checks to see if the origin of the event is a specific, critical node and raises the event severity if it is.

To create and activate this rule perform the following:

1. From the TME desktop double-click on the **Event Server** icon to display a window containing the rulebases.
2. Select the active rulebase with the right mouse button and select **Edit Rules** from the menu. See Figure 362 on page 385.

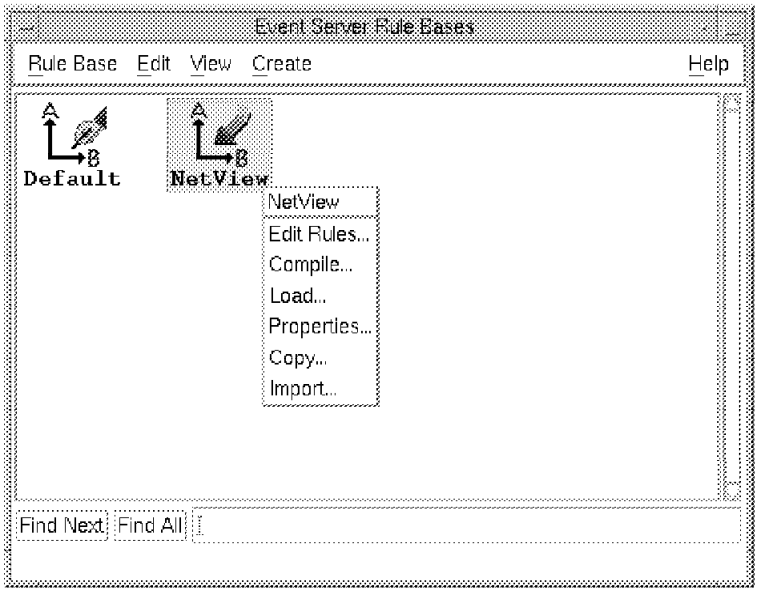


Figure 362. Edit Rules

3. Select **Ruleset** followed by **New Ruleset**. Figure 363 appears. (We entered the name Sysmon in the Set Name field, overwriting the name .new_set.)

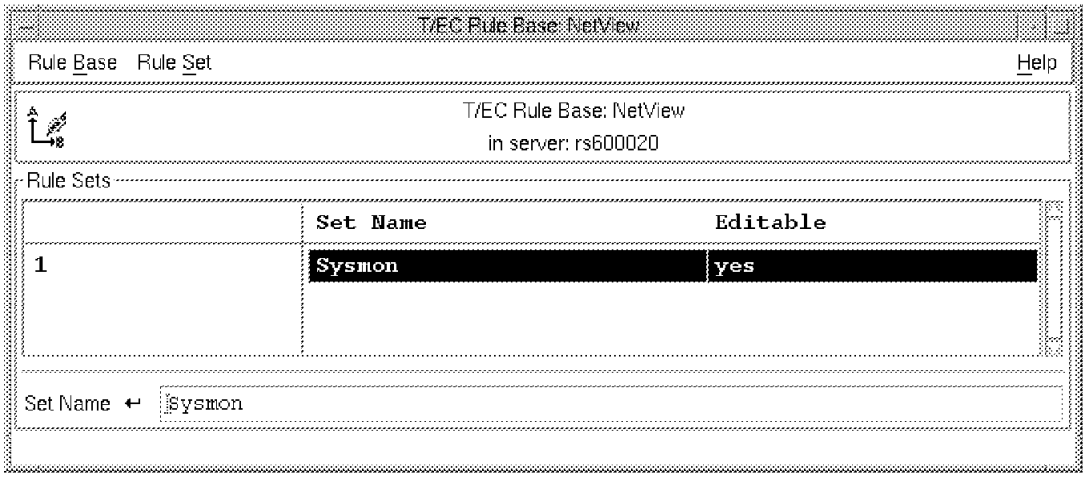


Figure 363. New Ruleset

4. Select the ruleset name and then select **Rule** followed by **New Rule**. Then choose **Simple Rule** from the menu bar as shown in Figure 364 on page 386.

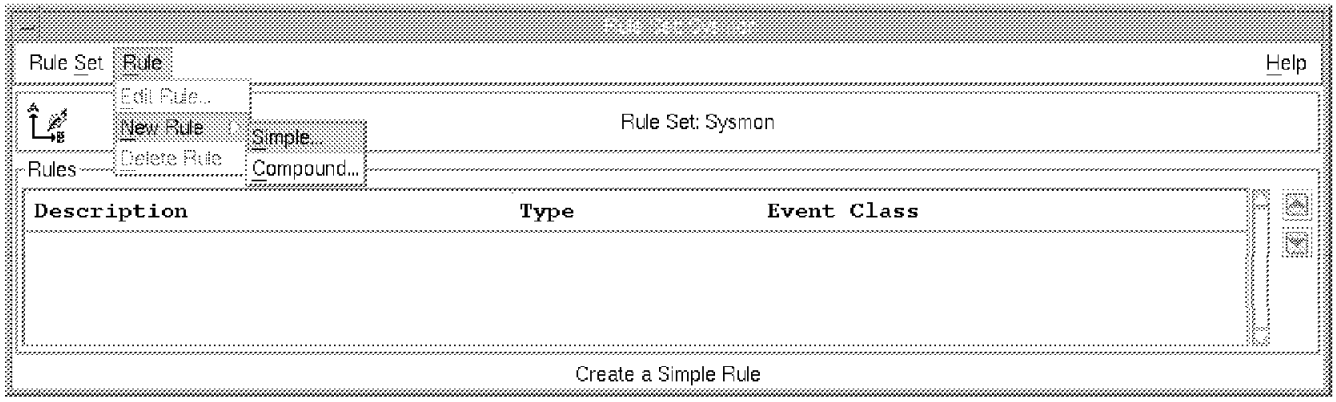


Figure 364. Creating the New Rule

5. Enter the description field and select the event class that will trigger the rule (in this case SM6K_ARM) by clicking on **Event Class**. The class must have already been imported into the rulebase.

6. Next define the further conditions under which the rule will be triggered. In our case we are taking action only if the event is from a particular critical node (IP address 9.24.104.28). Click on **Conditions** and enter the attribute values you want to check, as shown in Figure 365.

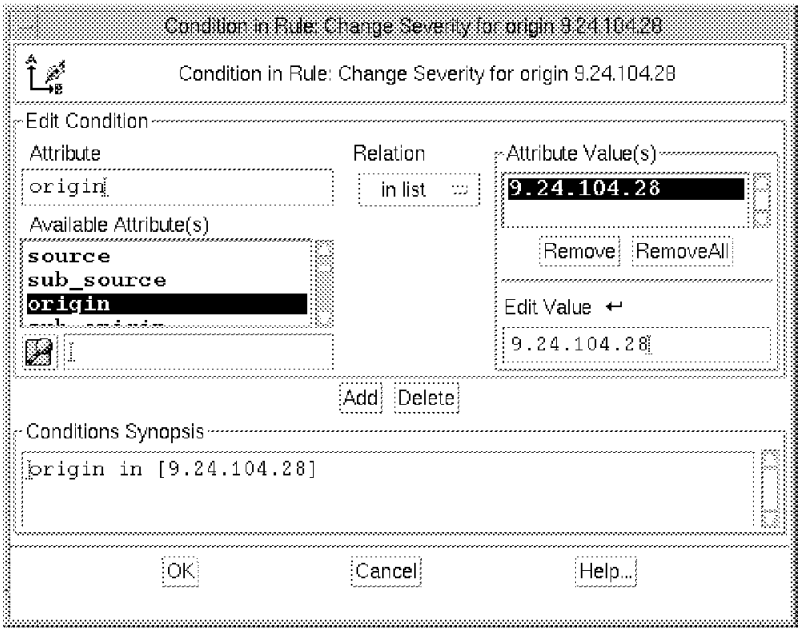


Figure 365. Entering Event Conditions

7. Return to the Simple Rule window and enter the actions to be taken. Figure 366 on page 387 shows the action that we took in this case, namely raising the event severity.

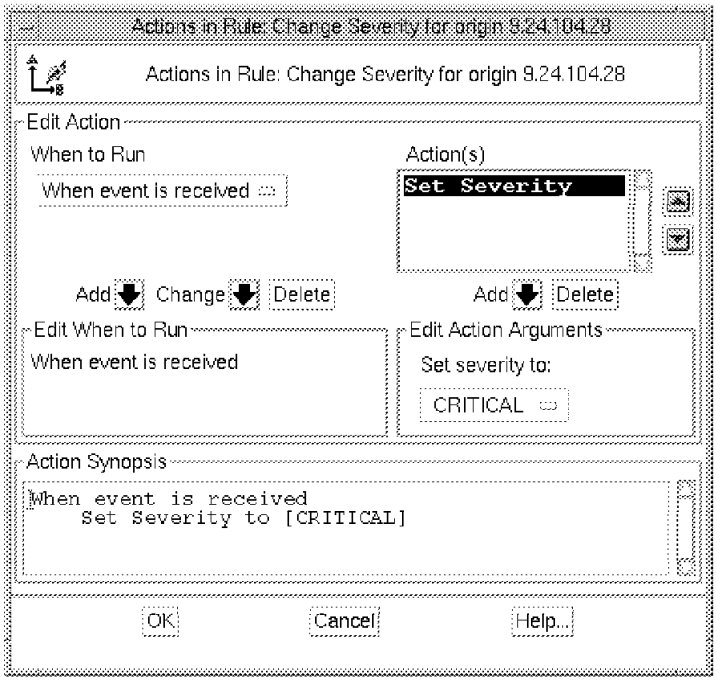


Figure 366. Action Definition

The completed configuration for this rule is shown in Figure 367.

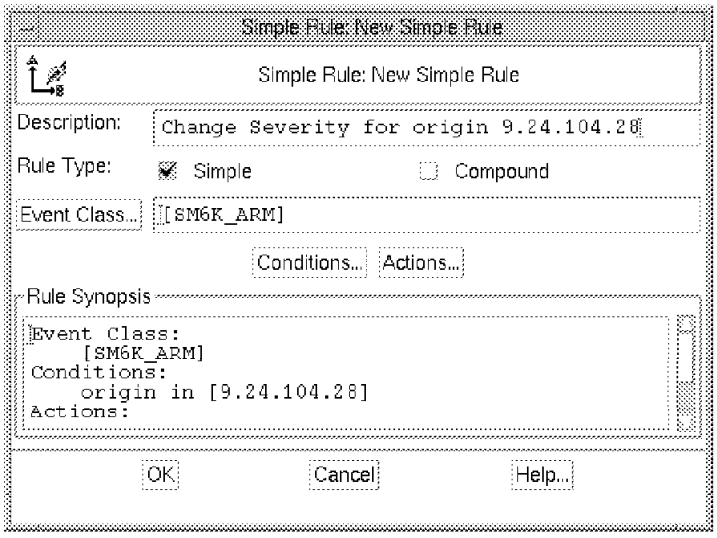


Figure 367. Rule Definition

After saving the ruleset we next must compile it. To do this, return to the ruleset display and select the following options (see Figure 362 on page 385).

1. Select **Compile** and enter N for the trace option.

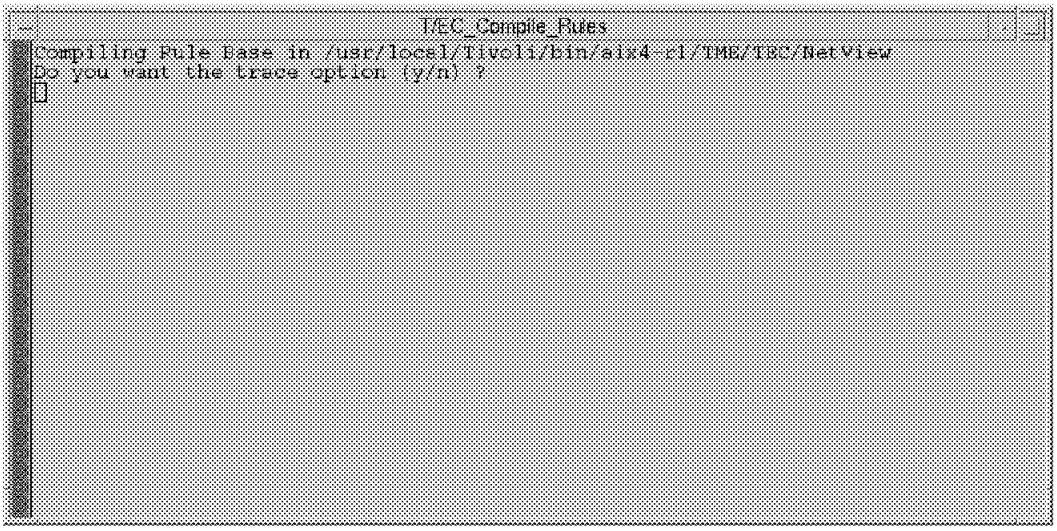


Figure 368. Compiling the Rulebase

2. Select **Load** and select the option shown in Figure 369.

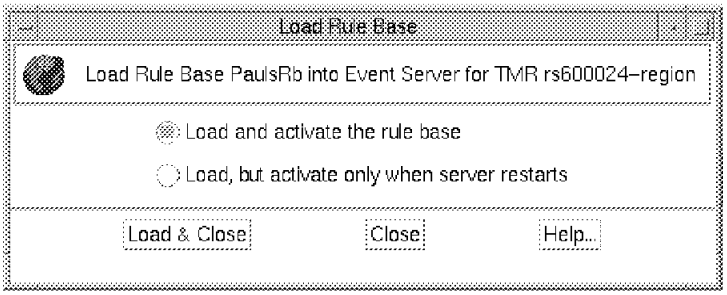


Figure 369. Load the Ruleset

Note that in this case the Enterprise server does not need to be restarted. You only have to restart the server after compiling and reloading the rulebase if you have added new event classes to it.

Figure 370 on page 389 shows the result of this rule being applied. The events have changed from WARNING to CRITICAL.

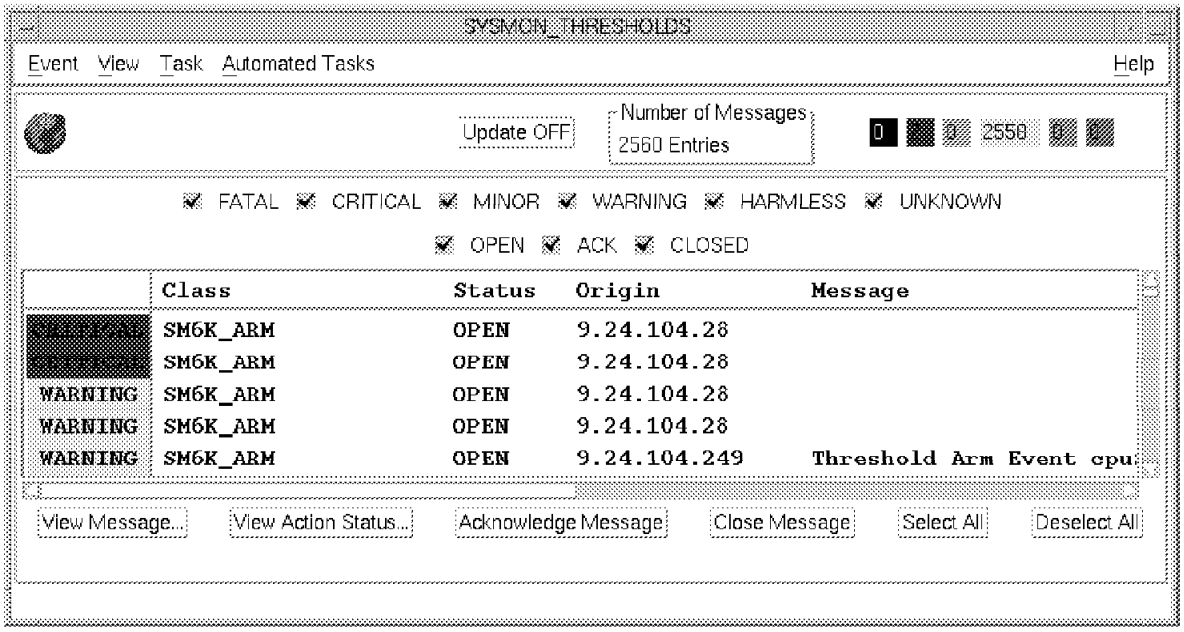


Figure 370. The Working Ruleset

12.1.3 Coding Event Rules Manually

The GUI is limited to creating and modifying simple rulesets. The example below shows how to create and activate a ruleset using your favorite UNIX editor. In this example we are working with events from the NetView ruleset adapter (the PTF U443133 enhancement). NetView sends Node Up and Node Down events to reflect the IP status of nodes in the network. We create a rule that will close the Node Down event for a specific machine if a Node Up event is received for the same machine within a fifteen minute period.

Figure 371 on page 390 shows the T/EC rule code.

```
rule: nv_up_down: (

     description: 'Correlate NVUP DOWN',

     event: _event of_class 'Nvserverd_Event'

     where     [
               nv_specific: equals 0x3830000,
               status: outside ['CLOSED'],
               nv_var2: _hostname
               ],

     action:     (
          first_instance(
               event: _old_event of_class 'Nvserverd_Event'
                   where [
                         nv_specific: equals 0x3830001,
                         status: outside ['CLOSED'],
                         nv_var2: equals _hostname
                         ],
                      _event - 900 - 60
          ),
          set_event_status(_event, 'CLOSED'),
          set_event_status(_old_event, 'CLOSED'),
          link_effect_to_cause(_old_event, _event)
          )
) .
```

Figure 371. Node Up/Down Rule Definition

Let us review the construction of this rule:

- The first two lines identify the rule name and give a brief description.
- The *event* line checks for events of class Nvserverd_Event (the NetView for AIX reulset adapter). It also assigns the event ID to ruleset variable _event.
- The *where* section checks the nv_specific and status slots to make sure the event matches the requirement and then assigns the contents of the nv_var2 slot to ruleset variable _hostname. (NetView internal events usually place the affected hostname in the second trap variable.)
- The *action* section contains four action lines. The first of these, *first instance*, searches back in time for a corresponding event. The search area spans back in time 900 seconds and forward in time 60 seconds (it may seem odd to search forward, but the rule processing may take some time to complete, during which time other matching events can arrive). If a match is found the following action lines are executed, therefore changing the status of both the original (Node Down) and the current (Node Up) event and also putting an indicator into one event to tie the events together.

There are a few parameters contained in the ruleset that are NetView-related. The class type Nvserverd_Event is defined in the nvserverd.baroc file which is part of the NetView ruleset adapter. The parameter called nv_specific is the NetView for AIX specific trap ID in hexadecimal format. The specific trap ID is usually quoted in decimal, so you need to convert it. One way to do this is to use the UNIX bc (binary calculator) command. The procedure is:

- From the NetView for AIX machine, find the specific trap IDs for Node Up and Node Down. You can use the NetView GUI or the `event -l` command, for example:

```
event -l | grep NUP_EV
NUP_EV       0058916864    3    Node Up
event -l | grep NDWN_EV
NDWN_EV      0058916865    3    Node Down
```

- Calculate hexadecimal values using bc. We entered the following command sequence to get the value for the Node Up trap:

  ```
  bc
  obase=16
  0058916864
  ```

The resulting hexadecimal values are 3830001 (Node Down) and 3830000 (Node Up). These two numbers are used in the rule to identify the Node Up and Node Down events.

12.1.3.1 Activating the Ruleset

To activate the ruleset you have to import it into the T/EC rulebase and then compile the rulebase. On the T/EC server machine rs600020, we performed the following steps to create and activate the rule:

1. Create the ruleset file /u/paul/tec_adapter_area/nvupdown.rls, as listed in Figure 371 on page 390.
2. Import the new ruleset into the NetView rulebase with the following command:

   ```
   wimprbrules /u/paul/tec_adapter_area/nvupdown.rls NetView
   ```

3. Compile the rulebase using the following command:

   ```
   wcomprules -t NetView
   ```

4. Load and activate the NetView rulebase:

   ```
   wloadrb -u NetView
   ```

You can also execute these functions from the rulebase context menu, by selecting the rulebase icon with the right mouse button. Be careful to follow the Prolog conventions for labels and variable names. Any name that begins with a capital letter or an underscore (_) is assumed to be a variable. For example, we originally named the rule Nvupdown. This caused the compilation to fail with the following message:

```
WARNING: Invalid rule set specification rule_set
 (Nvupdown ,'nvupdown.rls',active)
```

12.1.4 Tracing Rules

As soon as you move beyond very simple T/EC rules, you are likely to need to trace the rule processing. There are two steps to activate the tracing function:

1. The rulebase must be compiled with the -t option.
2. The Event Server tracing parameter must be set, by clicking on the **EventServer** icon with the right mouse button and selecting **Parameters**. Figure 372 on page 392 shows this dialog.

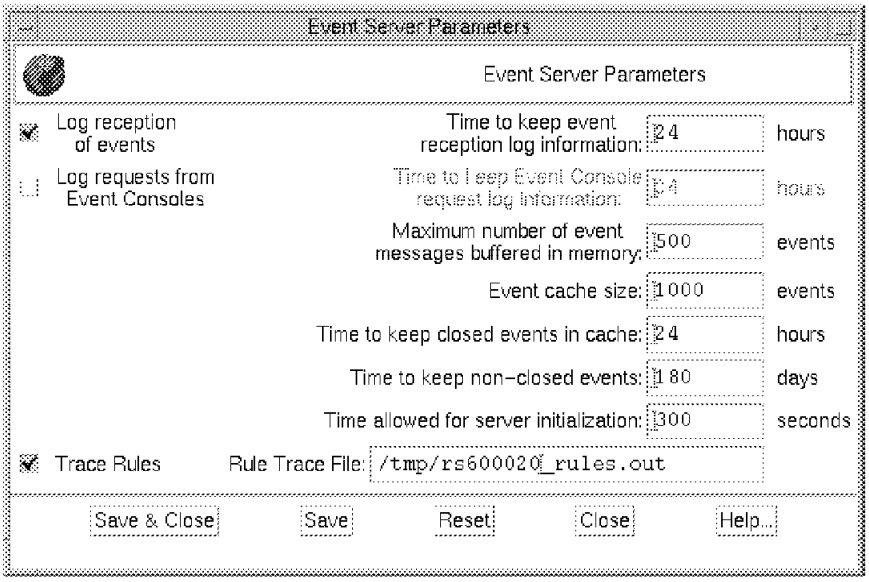


Figure 372. Activate Tracing

You have to stop and restart the T/EC server before these changes will take effect. We traced the NetView node up/down correlation rule described above. The resulting output is shown in Figure 373. You can see the condition and action sections of the rule being invoked. For each statement there is a *call* record in the trace. If the statement is successful (that is, if the event matches the condition or the action completes successfully) it terminates with an *exit* record. If the statement is unsuccessful it produces a *fail* record.

```
[703]     => rule set nvupdown
[704]        -> rule    nv_up_down
                event : 0x20265558 of_class Nvserverd_Event
[705]           call    condition
[706]              call    nv_specific :  _636
[707]              exit    nv_specific : 0x3830001
[708]              call    nv_specific : 0x3830001 equals 0x3830000
[709]              fail    nv_specific : 0x3830001 equals 0x3830000
[710]           fail    condition
[711]     => rule set nvupdown
[712]        -> rule    nv_up_down
                event : 0x202657c8 of_class Nvserverd_Event
[713]           call    condition
[714]              call    nv_specific :  _636
[715]              exit    nv_specific : 0x3830001
[716]              call    nv_specific : 0x3830001 equals 0x3830000
[717]              fail    nv_specific : 0x3830001 equals 0x3830000
[718]           fail    condition
[719]     => rule set nvupdown
[720]        -> rule    nv_up_down
                event : 0x20264c68 of_class Nvserverd_Event
[721]           call    condition
[722]              call    nv_specific :  _636
[723]              exit    nv_specific : 0x3830000
[724]              call    nv_specific : 0x3830000 equals 0x3830000
[725]              exit    nv_specific : 0x3830000 equals 0x3830000
[726]              call    status :  _1713
[727]              exit    status : OPEN
[728]              call    status : OPEN outside [CLOSED]
```

Figure 373. Rule Trace for nvupdown.rls

12.2 Extending the Logfile Adapter to Manage a Distributed Application

In this section we show a practical example of how the T/EC Logfile Adapter may be applied to a specific application problem. The application that we chose was MQ Series, the IBM middleware product for reliable asynchronous distribution of application function. We ran the MQ components on AIX systems, but you should note that the NT logfile adapter is very similar in operation to the UNIX adapter, so the same example is applicable to both environments.

12.2.1 Test Scenario

The main functional component of MQ Series is the *Message Queue Manager* (MQM). In our case we have MQMs running on AIX system rs600010 which has connections to MQMs on another AIX system, rs60003 and the Windows NT PC mqnt1. The inter-MQM connections are made up of logical channels. Each connection consists of a sender and receiver channel.

When the status of these channels changes, the Queue Manager writes a message into a logfile. We set up the T/EC Logfile Adapter to monitor the logfile for these messages, generate a Tivoli event and display it on the T/EC event console. As a further enhancement we created a ruleset that, if an MQM loses the connection to one of its communication partners, will try to restart the channel automatically.

12.2.2 Defining New Logfile Event Classes

The events that the logfile adapter sends to the T/EC server are structured messages, starting with the event class name and followed by a series of slot values, for example: `Su_Success;hostname=rs600020;...'`. Each slot contained in the message must be defined within a baroc file that has been imported into the rulebase.

Before we can define the new data type for the application-specific event, we have to know what kind of information it should contain. After this we can tell the adapter how to extract the data from the MQ Series logfile and how to assign them to the fields in our new event class.

The place to start, therefore, is the original source message. Now let us look at the logfile written from the Message Queue Manager. On AIX, MQM stores its error messages in a file named /var/mqm/qmgrs/<hostname>/errors/AMQERR01.LOG. Figure 374 on page 394 shows an excerpt from the logfile.

```
08/15/96    19:43:54  AMQ9002: Channel program started.

EXPLANATION: Channel program 'RS600010.TO.MQNT1' started.
ACTION: None
----------------------------------------------------------------------
08/15/96    20:19:34  AMQ9001: Channel program ended normally.

EXPLANATION: The channel is closing because of a request by the user.
ACTION: None.
----------------------------------------------------------------------
08/16/96    09:35:15  AMQ9999: Channel program ended abnormally.
EXPLANATION: Channel program 'MQNT1.TO.RS600010' ended abnormally.

ACTION: Look at previous error messages for channel program
----------------------------------------------------------------------
```

Figure 374. Messages of Interest in Logfile AMQERR01.LOG

We can immediately see that this logfile will cause us some problems, because the output of one error message is distributed over three lines. The Logfile Adapter parsing mechanism can not look into more than one line to create a new event. Therefore, we will start by concentrating on only the lines that contain the error code (strings that begin with AMQ) in the message text. This is because these lines contain the most important information.

We configure the Logfile Adapter to apply the following algorithm to the MQM messages:

1. Examine all new incoming error messages in file AMQERR01.LOG every 15 seconds.
2. Look for lines with the string AMQ9001, AMQ9001 or AMQ9999.
3. Extract the following information from the line:
 - Date
 - Time
 - Error index
 - Error message
4. Generate a different event class for each of these errors, and send the event to T/EC.

12.2.2.1 Creating the Baroc Definition

Before the adapter can create a new event we have to build the class definition for the three message queue events, because unknown events are rejected by the T/EC. Each class definition has a unique identifier and a list of attributes, or *slots*. We can use inheritance to build our Message Queue Events. In our example we create the class Logfile_MQS which inherits all slot definitions from the event class Logfile_Base. This class already exists because the logfile adapter has been installed. The class Logfile_Base itself is a subclass from the basic root class EVENT. Table 14 on page 395 illustrates how the slot definitions are inherited. The underlined lines in the Logfile_Base class highlight the slots that are redefined or added beyond the base class definition. If you want to view event classes in detail, use the command `wlsrb -d` to retrieve the directory name of your rulebase and look at the baroc files in subdirectory TEC_CLASSES.

Table 14. Inheritance of Slot Definitions	
Class EVENT in file root.baroc	**Class Logfile_Base in file tecad_logfile.baroc**
source: STRING; sub_source: STRING; origin: STRING; sub_origin: STRING; hostname: STRING; adapter_host: STRING date: STRING; status: STATUS, default=OPEN; severity: SEVERITY, default=WARNING; msg: STRING; msg_catalog: STRING; msg_index: INTEGER; repeat_count: INTEGER;	source: STRING, default=″LOGFILE″; sub_source: STRING , default=″LOGFILE″; origin: STRING; sub_origin: STRING , default= ″N/A″; hostname: STRING; adapter_host: STRING , default= ″N/A″; date: STRING; status: STATUS, default=OPEN; severity: SEVERITY, default=WARNING; msg: STRING; msg_catalog: STRING , default=″none″; msg_index: INTEGER , default=0; repeat_count: INTEGER , default=0; pid: STRING, default=″N/A″;

Because our Logfile_MQS event class is derived from Logfile_Base, we can use all the slots shown in Table 14. We therefore only have to define the elements that are unique to the MQ Series messages. Figure 375 shows the baroc file tecad_mqs.baroc that we created. You can see that there are four classes defined in the file: a base class and then one class for each of the three messages in which we are interested.

```
TEC_CLASS :
        Logfile_MQS ISA Logfile_Base
        DEFINES {
                sub_source: default= ″MQS″;
                msg_date: STRING, default=″″;
                msg_time: STRING, default=″″;
        };
END

TEC_CLASS :
        MQS_Channel_started ISA Logfile_MQS
        DEFINES {
                severity: default = HARMLESS;
        };
END

TEC_CLASS :
        MQS_Channel_ended_normally ISA Logfile_MQS;
END

TEC_CLASS :
        MQS_Channel_ended_abnormally ISA Logfile_MQS;
END
```

Figure 375. BAROC Class Definitions for MQ Failure Messages

12.2.2.2 Defining the MQS Baroc Format to the T/EC Server

Incorporating our new event class definitions into the rulebase is identical to the process we have seen several times already. Briefly the process is as follows (we have listed the CLI commands, but each function is also available from the GUI, see Chapter 10, “Introduction to the TME 10 Enterprise Console” on page 319 for details):

1. Find the name of the current loaded rulebase. From the GUI, double-click on the **EventServer** icon and look for the rulebase with a red arrow pointing to it, or from the command line enter:

```
wlscurrb
```

2. Validate your events with the previously loaded event classes, using the command:

   ```
   wchkclass tecad_mqs.baroc your_rb_name
   ```

3. Import the class definition into the rulebase. The command is:

   ```
   wimprbclass tecad_mqs.baroc your_rb_name
   ```

4. Compile the rulebase. The command is:

   ```
   wcomprules your_rb_name
   ```

5. Load the rulebase. The command is:

   ```
   wloadrb your_rb_name
   ```

6. Stop and restart the event server by issuing the following commands:

   ```
   wstopesvr
   wstartesvr
   ```

You can check that your classes are loaded by issuing the following command:

```
wlsrbclass your_rb_name | grep MQS
```

The result will be something like the following:

```
Logfile_MQS
MQS_Channel_started
MQS_Channel_ended_normally
MQS_Channel_ended_abnormally
```

12.2.3 Mapping Message Formats to Event Classes

In our baroc definition we have specified what the output of the logfile adapter will look like. Now we have to write the instructions for the logfile adapter, to tell it how to extract the information from the first line of each message and fill in the event slots.

The adapter uses its own format description language to do this. We need to use this language to define the mapping shown in Figure 376.

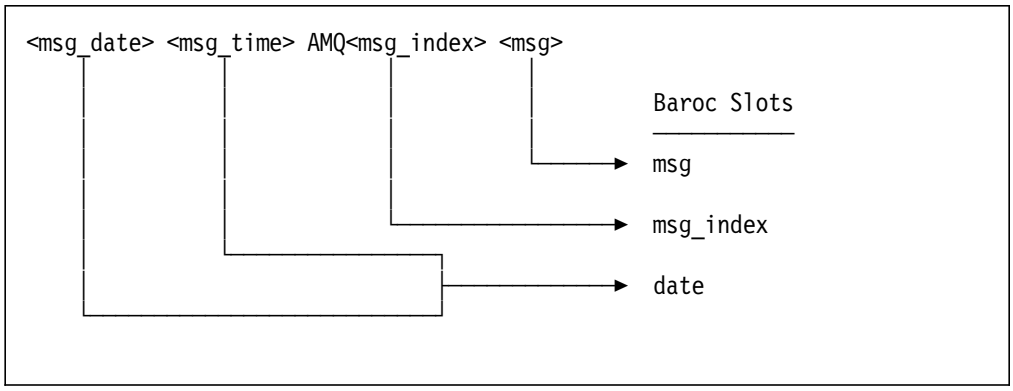


Figure 376. Mapping Logfile Messages to Baroc Slots. The msg and date slots are part of the root event class, msg_index is added by our own baroc definition.

For each new event class we appended a format instruction to the /etc/Tivoli/tecad/etc/tecad_logfile.fmt file. The first line in every format description contains a search pattern. The adapter is only able to map the message to an event class if the message line matches the search pattern. The search pattern uses the following parsing controls:

1. %s matches one string.
2. %s* matches zero or more strings separated by blank space.
3. %s+ matches one or more strings separated by blank space.
4. %t matches a time stamp with the month date time format.
5. Other characters are treated as constants.

Figure 377 shows the lines we added to the format file for the MQS events.

```
// Adapter format specification for Message Queue logfile
// stored in /var/mqm/qmgrs/RS600010/errors/AMQERR0?.LOG

FORMAT Logfile_MQS
%s %s AMQ%s: %s*
hostname DEFAULT
origin DEFAULT
msg_date $1
msg_time $2
date PRINTF("%s %s", msg_date, msg_time)
msg_index $3
msg $4
END

FORMAT MQS_Channel_started FOLLOWS Logfile_MQS
%s %s AMQ9002: %s*
msg_index 9002
msg $3
END

FORMAT MQS_Channel_ended_normally FOLLOWS Logfile_MQS
%s %s AMQ9001: %s*
msg_index 900
msg $3
END

FORMAT MQS_Channel_ended_abnormally FOLLOWS Logfile_MQS
%s %s AMQ9999: %s*
msg_index 9999
msg $3
END
```

Figure 377. Additional Format Description for the Logfile Adapter

Notice that the string AMQ represents a constant value. All other information in the message can be assigned to slots. Also, the time format in the MQ messages contains additional blank space, which is not what the logfile adapter expects. Therefore, we cannot use the %t specification to map the message time stamp. Instead, the format places the date and time components into two additional slots and then combines them to make the time slot.

12.2.4 Updating the Logfile Adapter Configuration Files

Although we have defined the mapping by updating the .fmt file, the logfile adapter does not use this file directly. Instead, it requires *class definition statements*, which are contained in the file tecad_logfile.cds. This file can be generated automatically from the format description file (tecad_logfile.fmt). In order to get the CDS, execute the following commands:

1. Call the generator from the logfile adapter:

   ```
   cd /etc/Tivoli/tecad/etc
   ../bin/logfile_gencds tecad_logfile.fmt
   ```

2. Check the output for warnings and syntax errors:

3. Create the CDS file with the command:

   ```
   ../bin/logfile_gencds tecad_logfile.fmt > tecad_logfile.cds
   ```

Before you restart the logfile adapter to pick up the changes to the .cds file, you also have to modify the main adapter configuration file (tecad_logfile.conf). This is necessary because the logfile adapter, by default, reads messages from Syslog. However, our application messages (MQM messages) are not written to the syslogd daemon, but to a private logfile. We must place the following line in /etc/Tivoli/tecad/etc/tecad_logfile.conf:

```
LogSources=/var/mqm/qmgrs/<hostname>/errors/AMQERR01.LOG.
```

Finally you can stop and restart the logfile adapter to pick up the changes:

```
../bin/init.tecad_logfile stop
../bin/init.tecad_logfile start
```

12.2.5 Debugging the Logfile Adapter

When you extend the logfile adapter as we have described it is possible that you will make some errors, so that the event will not be sent to the T/EC server. There are several points at which you can trace the activity. In order to switch on the trace you have to edit file /etc/Tivoli/tecad/etc/tecad_logfile.err. Each module of the adapter is represented with a section name and severity level. The adapter sends trace messages to the /dev/null device normally, but it is possible to redirect it to any file. You have to change only four lines to trace full details of the parsing process:

1. # MODULE = MAP
 MAP LOW /tmp/tecad_logfile.trace
2. # MODULE = TECIO
 TECIO LOW /tmp/tecad_logfile.trace
 TECIO FATAL /tmp/tecad_logfile.trace
 TECIO NORMAL /tmp/tecad_logfile.trace

After the logfile adapter has been restarted you can view the trace with the command `tail -f /tmp/tecad_logfile.trace`. First check whether the T/EC interface was initialized successfully. If you can find a message saying Cannot create EIF handle, then edit the configuration file /etc/Tivoli/tecad/etc/tecad_logfile.conf and check the ServerLocation entry. The adapter will be stopped if it cannot create the connection to the server.

Figure 378 on page 399 shows an excerpt from the logfile adapter trace file. You can see how the parser is mapping message field values into event slots and then the contents of the baroc message that is sent to the T/EC server. The first message contains the AMQ9002 message code (indicating a channel program start), and you can see that it is mapped correctly. The second message is the additional information about the channel for which our CDS file does not contain a mapping. It therefore maps to a class of Logfile_Base and the event is not sent because the adapter configuration file contains, by default, a filter definition to filter out such events (see 11.3.2.1, “The Adapter Configuration File” on page 362).

```
... LOW: MAP ... : Selected Map : <hostname> <rs600010>
... LOW: MAP ... : Selected Map : <origin> <9.24.104.109>
... LOW: MAP ... : Selected Map : <msg_date> <08/29/96>
... LOW: MAP ... : Selected Map : <date> <08/29/96 19:40:04>
... LOW: MAP ... : Selected Map : <msg_index> <9002>
... LOW: MAP ... : Selected Map : <msg> <Channel program started.>
... LOW: TECIO . : Sending event to T/EC ...
MQS_Channel_started;hostname=rs600010;origin=9.24.104.109; ... END
... LOW: TECIO.. : Event sent to T/EC

... LOW: MAP ... : Selected Map : <hostname> <'RS600010.TO.MQNT1'>
... LOW: MAP ... : Selected Map : <date> <EXPLANATION: Channel ...
... LOW: MAP ... : Selected Map : <origin> <???>
... LOW: MAP ... : Selected Map : <msg> <started.>
... LOW: TECIO . : Sending event to T/EC ...
Logfile_Base;hostname='''RS600010.TO.RS60003''';date='EXPLANATION ... END
... LOW: TECIO . : No event sent to T/EC
```

Figure 378. Debug Information from the Logfile Trace File

12.2.6 Preparing Event Consoles to Receive MQS Events

Because we want to collect all MQS events in one event group, we created the group MQS Log. The event from MQS and all other logfile events are distinguished in the slot name sub_source. Therefore, we check the filter for the constant MQS.

1. Create a new event group.



Figure 379. Create a New Event Group MQS Log

2. Define a filter for the event group MQS Log.

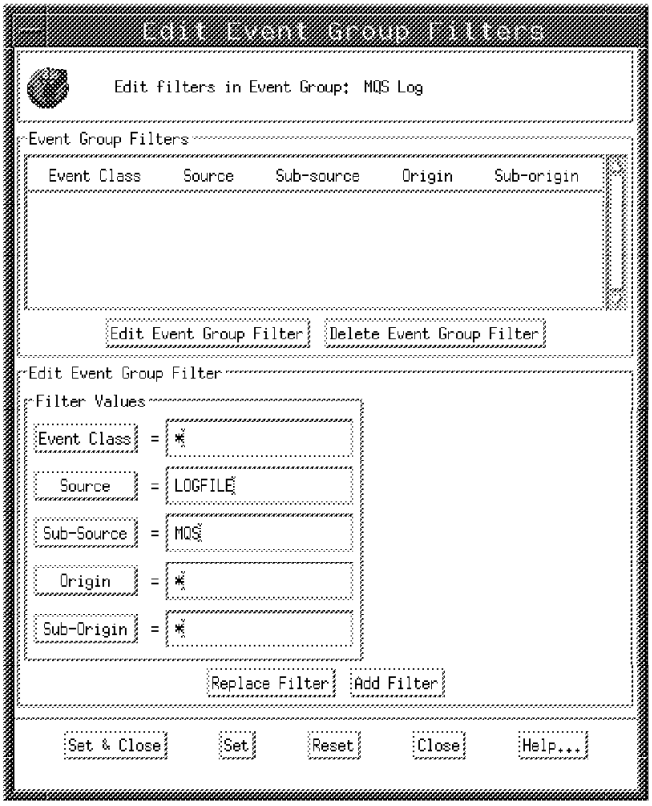


Figure 380. Define a Filter for the Event Group MQS Log

3. Assign the event group to the event console.

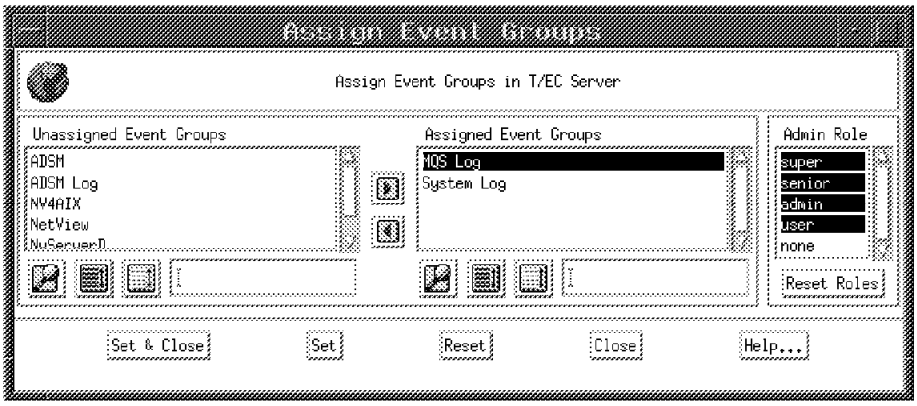


Figure 381. Assign the Event Group MQS Log to Your Event Console

12.2.7 Testing Events from MQ Manager on the T/EC

To test the operation of our extended logfile adapter we used the MQ Series command line interpreter to change the state of the static connection between hosts rs600010 and MQNT1. Figure 382 on page 401 shows the dialog.

```
[root@rs600010] # runmqsc
5765-115 (C) Copyright IBM Corp.  1994.  ALL RIGHTS RESERVED.
Starting MQSeries Commands.

start channel(rs600010.to.mqnt1)
     1 : start channel(rs600010.to.mqnt1)
AMQ8018: Start MQSeries channel accepted.
5765-115 (C) Copyright IBM Corp.  1994.  ALL RIGHTS RESERVED.
Channel program started.
stop channel(rs600010.to.mqnt1)
     2 : stop channel(rs600010.to.mqnt1)
AMQ8019: Stop MQSeries channel accepted.
AMQ9528: User requested closure of channel 'RS600010.TO.MQNT1'.
Channel program ended normally.
```

Figure 382. Start and Stop the Static Channel

The events resulting from these commands are shown in Figure 383 and the details of one of the messages is shown in Figure 384 on page 402.

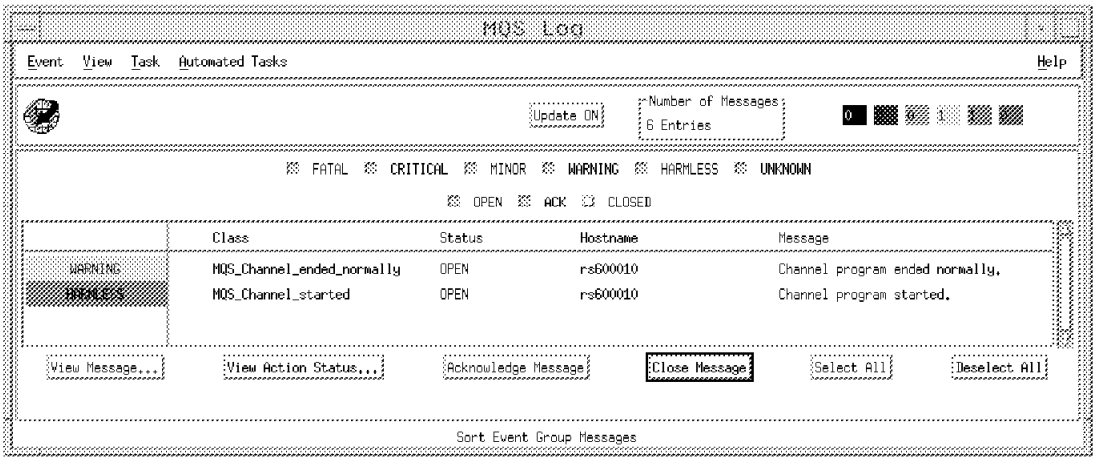


Figure 383. Events Generated from T/EC Logfile Adapter

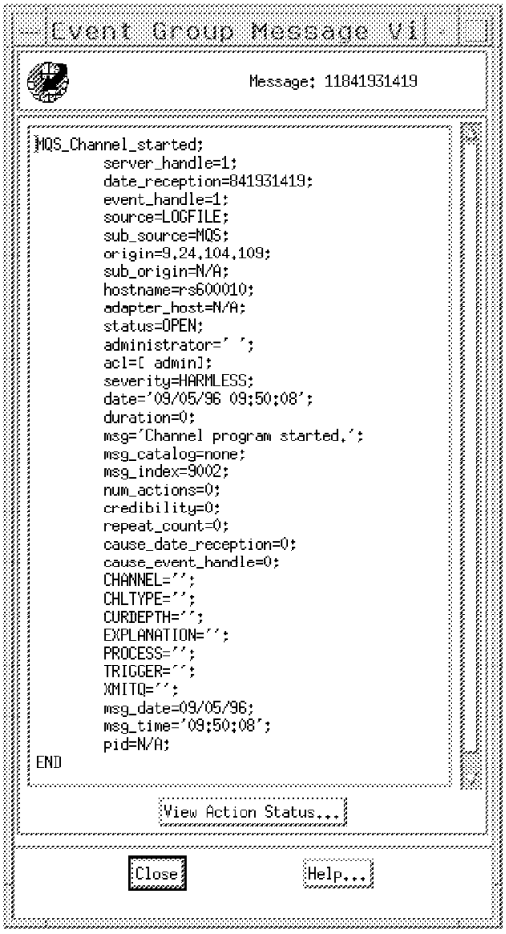


Figure 384. Event MQS_Channel_Channel_started

12.2.8 Create a Task to Restart the Channel

We have now achieved the first objective: awareness of MQ Series events from an event console. However, it would be much more useful if we could start an action to resolve the problem remotely.

With a TME task we can execute a shell script on the machine where the event occured (that is, where the MQ Manager wrote the message into its logfile). We created a script that uses the command line interface of MQ Manager to get information about the channel in order to restart it. The following list describes the functionality of the shell script:

1. Extract the channel name from the MQ logfile

 Restarting a channel is only possible if the channel name is known. This information is contained in the original MQ message, which is stored in the MQ logfile, but not in the Tivoli event. This is because of the limitation that the logfile adapter cannot extract values from a multiple message line. Therefore, the shell script has to read the MQ logfile again to get the channel name. The shell script can find the correct message based on the following fields:

 - Date
 - Time

- Message index

These fields *are* contained in the T/EC event and can be passed to the script as arguments.

2. Getting information about the channel.

 Using the MQM runmqsc command we are able to read more information about the channel and the queues.

3. Restart the channel if necessary.

 In case of the static connection between rs600010 and MQNT1 the channel can be started immediately. In case of a dynamic connection, such as the one between rs600010 and rs60003, the script checks first whether a particular channel flag is set and then tries to reestablish the channel. If it is not possible to change the flag or reestablish the channel, the script sends a new event to the T/EC:

 - MQS_Queue_Trigger_failed if the flag TRIGGER has not been set.
 - MQS_Channel_Ping_failed if the channel can't be reestablished.

The shell script is listed in Figure 385.

```
#! /bin/ksh
#####################################################################
#####################################################################
##
##  Filename: TEC_Start_MQS_Channel.ksh
##
##  Date of the last modification: 09/02/96
##
##  Description: Korn-shell script to restart a channel of
##               Message Queue Series.
##
##  Arguments:   ·-d trace_file] message_date message_time message_index
##
##  Authors:     Guenther Mayerhoffer
##
#####################################################################
#####################################################################

# defining some constants

  MQS_LOGFILE=/var/mqm/qmgrs/RS600010/errors/AMQERR01.LOG
  EXIT_USAGE=1
  EXIT_NOTFOUND=2
  EXIT_TRIGGER=3
  EXIT_PING=4

# setting debug flag for 'echo' statements

  DEBUG=""
  if [ $# -ge 2 ]
  then if [ "$1" = "-d" ]
       then
            shift
            [ ! -z $1 ] && DEBUG=">> $1"
            shift
       fi
  fi
```

Figure 385 (Part 1 of 4). TEC_Start_MQS_Channel.ksh Shell Script

```
# test number of arguments

  if [ $# -ne 3 ]
  then # Arguments are missing
      eval "echo 'Usage: $0 [-d trace_file] message_date message_time message_index' $DEBUG"
         exit $EXIT_USAGE
  else # Assign arguments to variables
      DATE=$1
      TIME=$2
      MSG_INDEX=$3
  fi

# Searching in the MQS logfile for the error message

  eval "echo 'Searching for <$DATE $TIME AMQ$MSG_INDEX:> in the' \
       'logfile $MQS_LOGFILE' $DEBUG"

  LINE=grep -n "$DATE *$TIME *AMQ$MSG_INDEX:" $MQS_LOGFILE

# Get the line number

  [ -z "$LINE" ] && { echo "Error: No line in logfile $MQS_LOGFILE" \
                           "matches with <$DATE $TIME AMQ$MSG_INDEX:>"; \
                  exit $EXIT_NOTFOUND; }

  echo $LINE | awk -F ":" '{print $1}' | read LINE; LINE=expr $LINE + 2

# Get the error explanation from MQ logfile

  CHANNEL=tail -n +$LINE $MQS_LOGFILE | \
  { read EXPLANATION; echo $EXPLANATION; } | \
  awk -F "\'" '{print $2}'

# Getting information about the channel

  echo "DISPLAY QMGR QMNAME" | runmqsc | \
  while read OUTPUT; \
  do \
  case "$OUTPUT" in \
      QMNAME\(*) QMNAME=echo $OUTPUT | tr ')' '(' | awk -F "(" '{print $2}';; \
  esac \
  done

# Getting information about the channel

  echo "DISPLAY CHANNEL($CHANNEL) XMITQ CHLTYPE" | runmqsc | \
  while read OUTPUT; \
  do \
  case "$OUTPUT" in \
      XMITQ\(*)    XMITQ=echo $OUTPUT | tr ')' '(' | awk -F "(" '{print $2}';; \
      CHLTYPE\(*)  CHLTYPE=echo $OUTPUT | tr ')' '(' | awk -F "(" '{print $2}';; \
  esac \
  done

# Getting information about the queue

  echo "DISPLAY QUEUE($XMITQ) PROCESS CURDEPTH TRIGGER" | runmqsc | \
  while read OUTPUT; \
  do \
  case "$OUTPUT" in \
      PROCESS\(*)  PROCESS=echo $OUTPUT | tr ')' '(' | awk -F "(" '{print $2}';; \
      CURDEPTH\(*) CURDEPTH=echo $OUTPUT | tr ')' '(' | awk -F "(" '{print $2}';; \
      TRIGGER)     TRIGGER=TRIGGER;; \
      NOTRIGGER)   TRIGGER=NOTRIGGER;; \
  esac \
  done
```

Figure 385 (Part 2 of 4). TEC_Start_MQS_Channel.ksh Shell Script

```
# Restart Channel if necessary

  if [ "$PROCESS" = " " ]
  then # restart static connection

         eval "echo 'Starting channel <$CHANNEL> ...' $DEBUG"
         echo "START CHANNEL($CHANNEL)" | runmqsc

  else # dynamic connection

        if [ $TRIGGER = "NOTRIGGER" ]
        then # trigger flag must be set

               eval "echo 'Changing trigger flag of queue <$XMITQ> ...' $DEBUG"
               echo "ALTER QL($XMITQ) TRIGGER" | runmqsc

             # Test the trigger flag again

               echo "DISPLAY QUEUE($XMITQ) TRIGGER" | runmqsc | \
               while read OUTPUT; \
               do \
                 case "$OUTPUT" in \
                     TRIGGER)     TRIGGER=TRIGGER;; \
                     NOTRIGGER)   TRIGGER=NOTRIGGER;; \
                 esac \
               done

            postemsg -S rs600020 \
               -m "hostname $QMNAME $XMITQ Rule Engine changed NOTRIGGER to $TRIGGER" \
               "hostname=hostname" \
               "CHANNEL=$CHANNEL" \
                     "XMITQ=$XMITQ" \
                     "CHLTYPE=$CHLTYPE" \
                     "PROCESS=$PROCESS" \
                     "CURDEPTH=$CURDEPTH" \
                     "TRIGGER=$TRIGGER" \
               MQS_Queue_Trigger_changed LOGFILE

            eval "echo 'Event MQS_Queue_Trigger_changed has been sent to T/EC' $DEBUG"

               if [ "$TRIGGER" = "NOTRIGGER" ]
               then # trigger flag could not be changed

                      eval "echo 'Error: Trigger flag could not be changed' $DEBUG"
                      exit $EXIT_TRIGGER
               fi
         fi

         if [ $CHLTYPE = "SDR" ]
         then # Test availability of the channel

                eval "echo 'Send ping to channel $CHANNEL' $DEBUG"

                echo "PING CHANNEL($CHANNEL)" | runmqsc | \
                while read OUTPUT; \
                do \
                  case "$OUTPUT" in \
                      AMQ*) EXPLANATION=$OUTPUT;; \
                  esac \
                done

                case "$OUTPUT" in
            AMQ8019*) eval "echo 'Ping to channel $CHANNEL was succesfully' $DEBUG"
                  ;;
            *) # Ping was not succesfully, send event to T/EC
```

Figure 385 (Part 3 of 4). TEC_Start_MQS_Channel.ksh Shell Script

```
                    postemsg -S rs600020 \
                    -m "hostname $QMNAME $XMITQ Rule Engine cannot ping channel $CHANNEL" \
                    -r 'CRITICAL' \
                    "hostname=hostname" \
                               "CHANNEL=$CHANNEL" \
                             "XMITQ=$XMITQ" \
                             "CHLTYPE=$CHLTYPE" \
                            "PROCESS=$PROCESS" \
                            "CURDEPTH=$CURDEPTH" \
                            "TRIGGER=$TRIGGER" \
                            "EXPLANATION=$EXPLANATION" \
                             MQS_Channel_Ping_failed LOGFILE

                    eval "echo 'Event MQS_Channel_Ping_failed has been sent to T/EC' $DEBUG"

                 exit $EXIT_PING
                 ;;
          esac
        fi

        if [ $CURDEPTH -gt 0 ]
        then # establish the channel for waiting messages

               echo "Starting channel <$CHANNEL> ..." >> /tmp/mqs.log
               echo "START CHANNEL($CHANNEL)" | runmqsc
        fi
  fi
```

Figure 385 (Part 4 of 4). TEC_Start_MQS_Channel.ksh Shell Script

The shell script uses new events to document an error situation. These events must be defined in the class definitions and imported into the T/EC server. Therefore, we extended the baroc file for MQ events by adding the MQS_Queue_Trigger_failed and MQS_Channel_Ping_failed events and also added more slots, because using the script we have access to more application-specific information. These extensions can be used for advanced event correlation in rulesets.

```
TEC_CLASS :
        Logfile_MQS ISA Logfile_Base
        DEFINES {
                sub_source: default= "MQS";
                msg_date: STRING, default="";
                msg_time: STRING, default="";
                CHANNEL: STRING, default="";
                XMITQ: STRING, default="";
                CHLTYPE: STRING, default="";
                PROCESS: STRING, default="";
                CURDEPTH: STRING, default="";
                TRIGGER: STRING, default="";
                EXPLANATION: STRING, default="";
        };
END

TEC_CLASS :
        MQS_Channel_started ISA Logfile_MQS
        DEFINES {
                severity: default = HARMLESS;
        };
END

TEC_CLASS :
        MQS_Channel_ended_normally ISA Logfile_MQS;
END

TEC_CLASS :
        MQS_Channel_ended_abnormally ISA Logfile_MQS;
END

TEC_CLASS :
        MQS_Channel_Ping_failed ISA Logfile_MQS;
END

TEC_CLASS :
        MQS_Queue_Trigger_changed ISA Logfile_MQS;
END
```

Figure 386. Extended Class Definitions for MQ Failure Messages

Having written the recovery script we can create the TME task to invoke it. The following figures show how to create the task (refer to Chapter 4, "Task Libraries, Tasks and Jobs" on page 81 for details of the TME task capability).

1. Open the Policy Region TEC25Region.

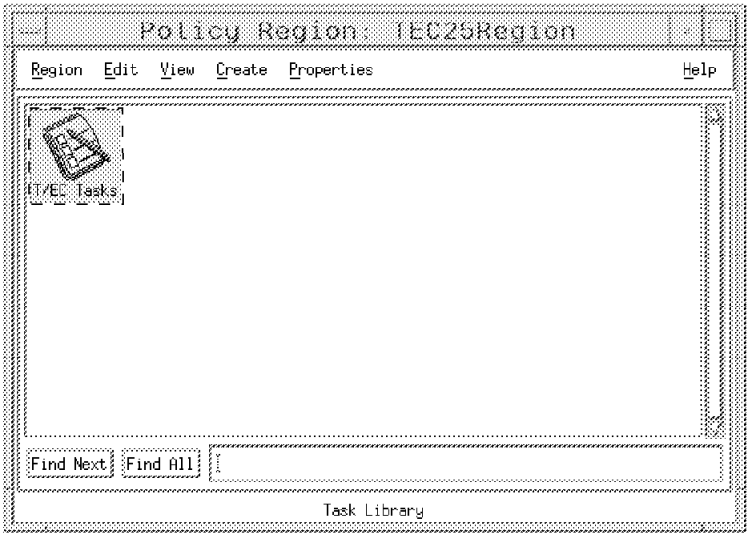


Figure 387. Default Policy Region for T/EC

2. Open the Task 'T/EC Tasks' and create a new Task.

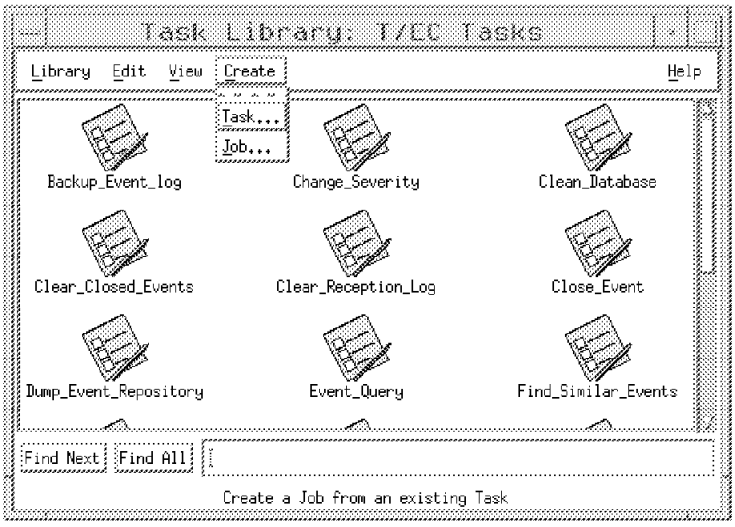


Figure 388. Default Task Library T/EC Tasks

3. Define the MQS Task.

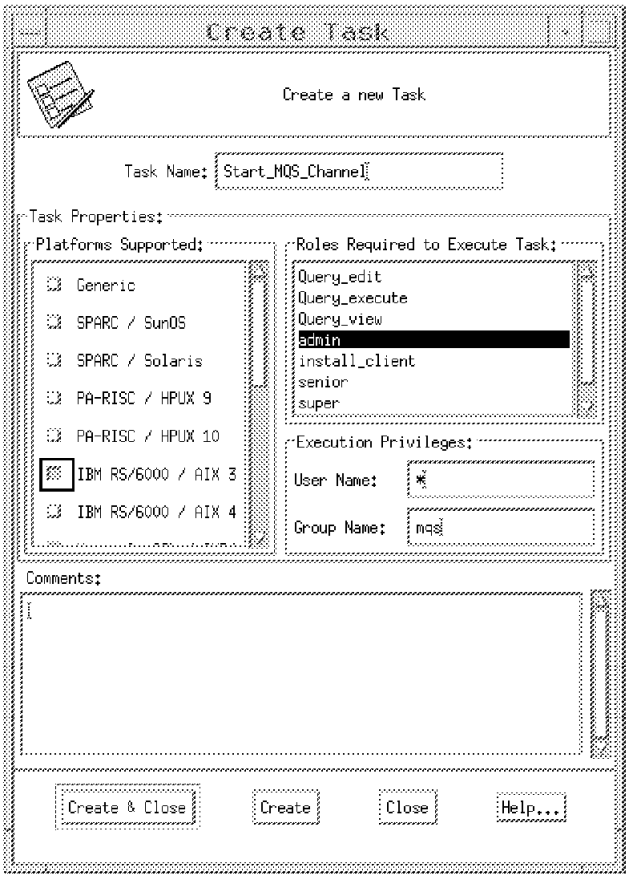


Figure 389. MQS Task Start_MQS_Channel

4. Enter the platform-specific shell script name for MQS Manager.

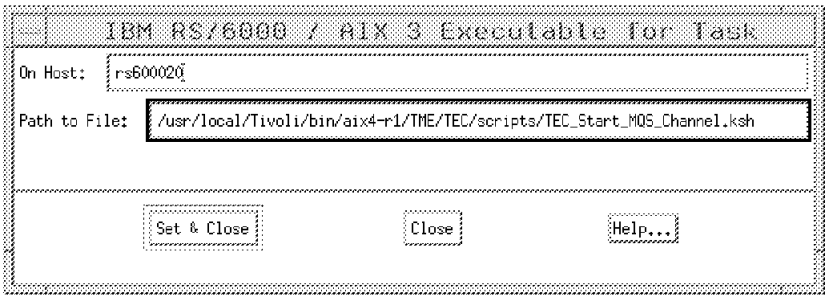


Figure 390. Executable for the AIX 3.2.5 Platform

12.2.8.1 Testing the Task

We cannot execute the new task directly from the desktop because the shell script expects some arguments. It is possible to provide arguments to a TME task from the desktop, but you cannot use the standard dialog. In 15.1, “Using the Task Library Language (TLL)” on page 455 we show an example of creating a task with arguments. However, for the purpose of testing our current task we can use the wruntask command to test the functionality of the script, for example:

```
wruntask   -l "T/EC Tasks" \
 -t "Start_MQS_Channel" \
 -h rs600010 \
 -a -d \
 -a "/tmp/Start_MQS_Channel.log" \
 -a "08/16/96" \
 -a "09:35:15" \
 -a "9999"
```

The arguments -d and /tmp/Start_MQS_Channel.log force the script file to write messages into the given file instead of the console.

12.2.9 Executing a TME Task from a T/EC Ruleset

Instead of executing the task manual we intend to restart the channel automatically using the rule engine. The action template to execute a TME task is *exec_task*. Figure 391 shows the rule that we used to trigger automatic recovery.

```
rule:
channel_ended_abnormally :
(
  event: _event of_class 'MQS_Channel_ended_abnormally'
         where [
                  status: _status equals 'OPEN',
                  hostname: _hostname,
                  msg_date: _msg_date,
                  msg_time: _msg_time,
                  msg_index: _msg_index
                ],
  reception_action: restart_channel:
  (
    exec_task( _event,
               'Start_MQS_Channel',
               '-l "T/EC Tasks" -h "%s" -a %s -a %s -a %d',
               [ _hostname, _msg_date, _msg_time, _msg_index ],
               'YES' )
  )
)
```

Figure 391. T/EC Rule to Start MQ Channel Recovery Automatically

To use this rule we have to import it into the active rulebase, compile the rulebase and reload it. Refer to 12.1, “Creating Enterprise Console Rulesets” on page 383 for an explanation of this process. From the command line enter the following commands:

```
wimprbrules tecad_mqs.rls your_rule_base_name
wcomprules
wloadrb -u your_rule_base_name
```

12.2.9.1 Take Care When Adding Rules with exec_task Actions

Our first version of the channel_ended_abnormally rule contained a small syntax error in the exec_task action, which had a major effect on the T/EC server. Instead of using a format field of %d, indicating a numeric field, for the _msg_index argument we used the string format identifier %s. Although the rules compiled correctly, whenever an event arrived in the T/EC server that matched the rule (and so executed the exec_task template) the following happened:

1. Several T/EC server processes failed:
 - Reception Engine (tec_reception)
 - Rules Engine (tec_rule)
 - Dispatch Engine (tec_dispatch)
 - Task Engine (tec_task)
2. Event consoles lost their connection to the T/EC server.

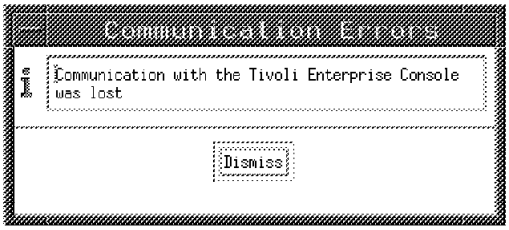


Figure 392. T/EC Server is not Available for Event Consoles

The master server from T/EC was still running but could not restart the processes. When we looked at the reception log using the wtdumprl command we could see that the status of the MQS_Channel_ended_abnormally event was QUEUED. This means that the rule engine could not process the event (Figure 393 shows the wtdumprl output).

```
1~ 14448~ 0~ 841418148(Aug 30 11:15:48 1996)
### EVENT ###
MQS_Channel_ended_abnormally;hostname=rs600010;origin=9.24.104.109;...
### END EVENT ###
QUEUED~' '
```

Figure 393. Unprocessed Event in the T/EC Reception Log

We found that we could stop and start the server, but the subprocesses and event console would not run because the T/EC rule again tried to process the MQS_Channel_ended_abnormally event. In fact, even if we deleted all events using the commands wtdbresetlog and wtdbclear -let 0, the problem persisted. Queued events cannot be deleted with these commands. The only way to recover was to remove the failing rule, recompile and reload the rulebase and then restart the event server.

12.2.10 Results of MQ Series Message Automation

We tested the automatic channel recovery by breaking a dynamic channel connection. Figure 394 shows the resulting sequence of messages. First the channel is started and then it fails abnormally. The MQS_Queue_Trigger_changed event tells us that the automatic recovery has changed the TRIGGER flag and is attempting a restart. Finally the MQS_Channel_Ping_failed event indicates that the recovery was unsuccessful. Figure 395 on page 412 and Figure 396 on page 413 show the slot contents of these last two events. Notice that the reason for the failure is in the EXPLANATION slot of the event.

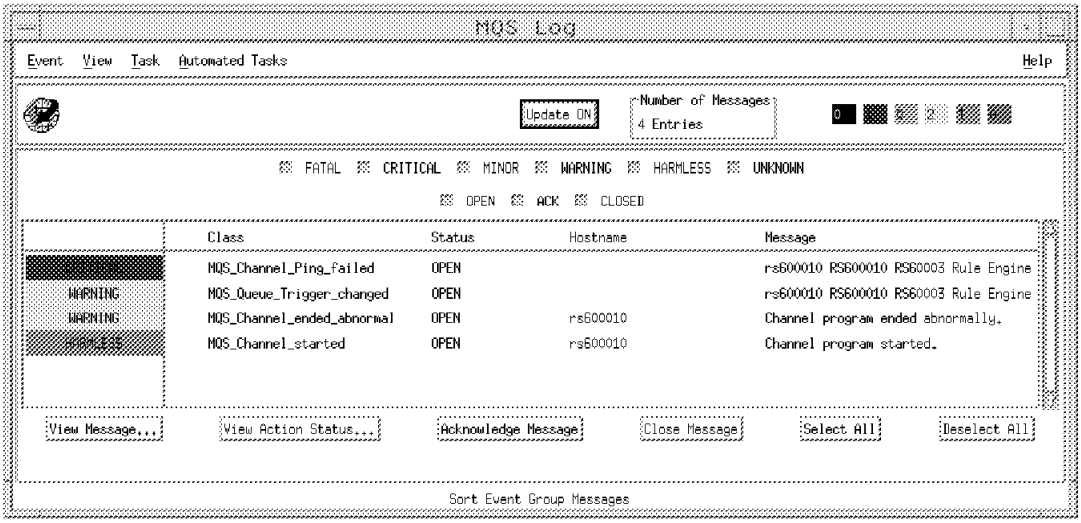


Figure 394. Events Generated through Executing Task from Rule Engine



Figure 395. Event MQS_Channel_Trigger_changed

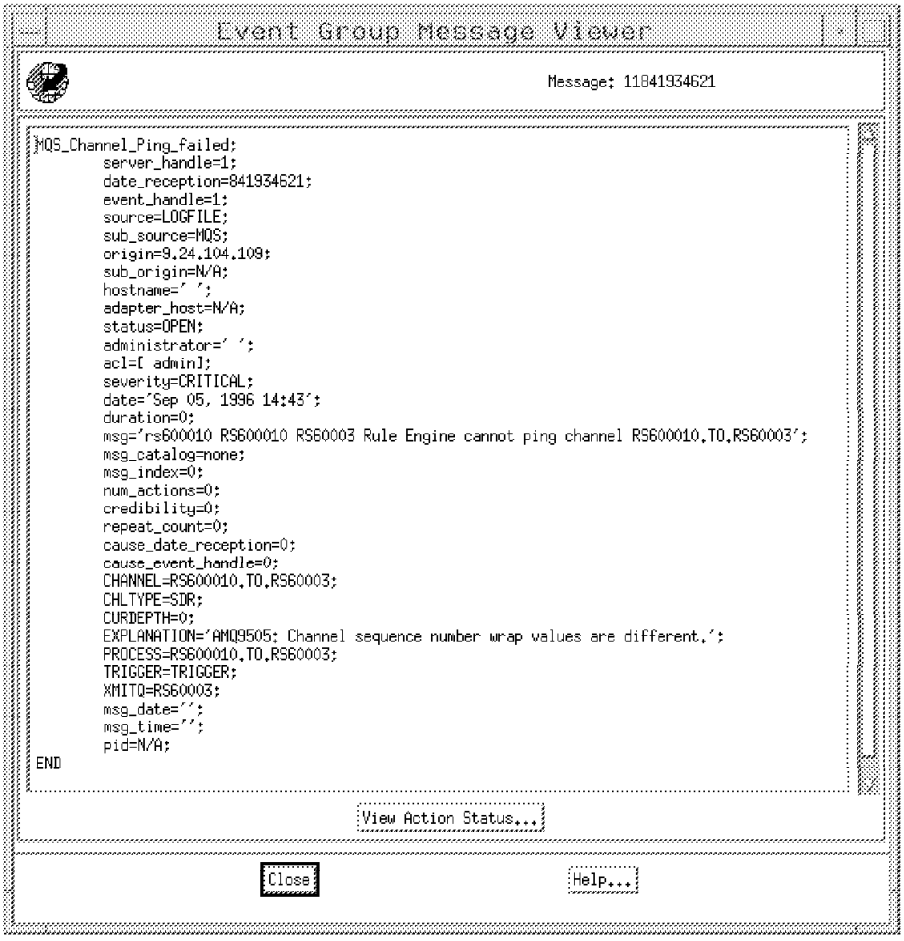


Figure 396. Event MQS_Channel_Ping_failed

12.3 Extending the NetView for AIX/OpenView Adapters

The NetView for AIX and Openview event adapters are predefined by default to handle a number of common traps and internal events. However, if you introduce other trap sources into your network you will have to modify the configuration of the adapter to make it forward the new events. Note that the new NetView for AIX ruleset adapter does not have this limitation; it simply maps the complete event contents into baroc event slots.

To illustrate the customization process we chose to integrate the threshold traps that are generated by the Systems Monitor Mid-Level Manager (MLM). However, the same process can be applied to any trap. For example, traps generated by network components such as hubs, routers and switches would have to be integrated using the same technique. At the end of this section we show how to incorporate additional trap types using the NetView for AIX ruleset adapter.

The examples involves modifying the T/EC event descriptions and behavior, creating new traps within NetView and showing how to define these traps as events within the T/EC.

12.3.1 Systems Monitor MLM Integration

The Mid-Level Manager provides a capability to disperse the polling load of an SNMP manager to a distributed system. It can poll MIB values and compare the results against thresholds. It also can act as a trap filter and can perform status polling on behalf of NetView for AIX. MLM uses SNMP traps to inform the SNMP manager of problems or changes that it detects.

First we had to identify the relevant Sysmon MIB definitions that are passed as arguments within the SNMP traps. The MIB definitions are contained in the file /usr/OV/snmp_mibs/ibm-sysmon-mlm.mib. However, the file only contains the ASN.1 definition for the MIB. When a trap arrives it contains MIB object IDs in dotted decimal form, but we want to use more meaningful object names in the T/EC event class definition. The NetView and Openview adapters provide a mapping file, /etc/Tivoli/tecad/etc/tecad_nv6k.oid, to interpret between dotted decimal object IDs and object names.

12.3.1.1 Obtaining MIB Object IDs from an SNMP Trap

Our first problem was to find out which MIB objects were passed within the MLM *Threshold Arm Event* trap. We used the process described below to do this.

Within NetView for AIX we changed the formatting of the the trap in order to see what arguments are passed by the application. Figure 397 on page 415 shows the Modify Event dialog in NetView for AIX. To list the complete contents of the event, we inserted the following format string at the start of the Event Log Message field:

```
Event Arguments $*\n
```

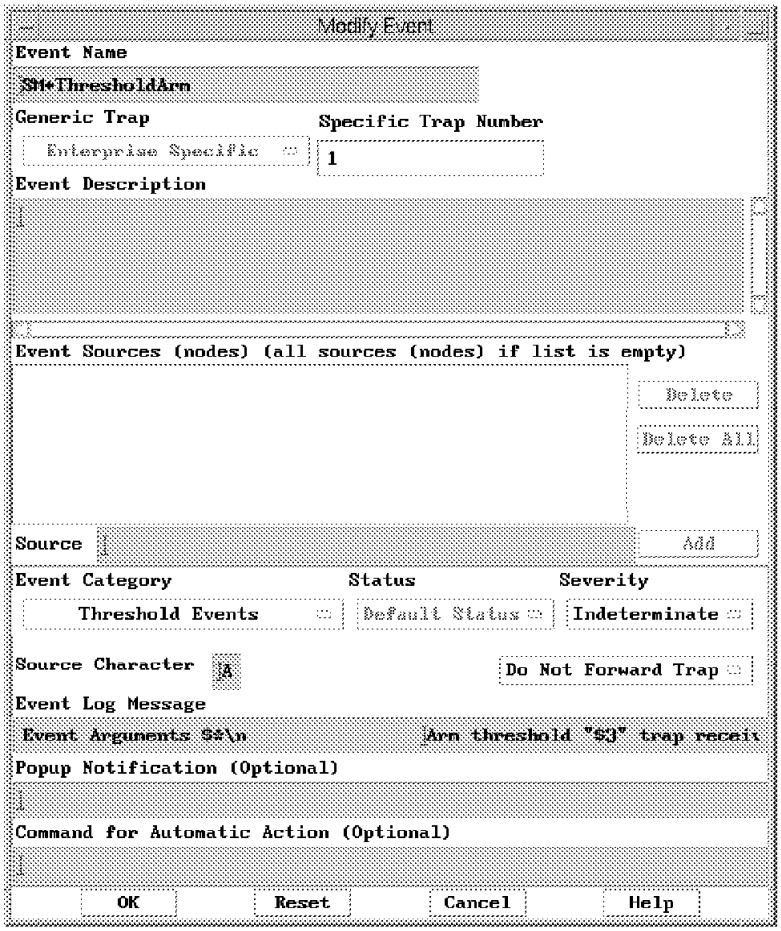


Figure 397. Systems Monitor Event Description

The $* construct causes all of the MIB variables in the event to be displayed when the event arrives, as shown in Figure 398 on page 416.

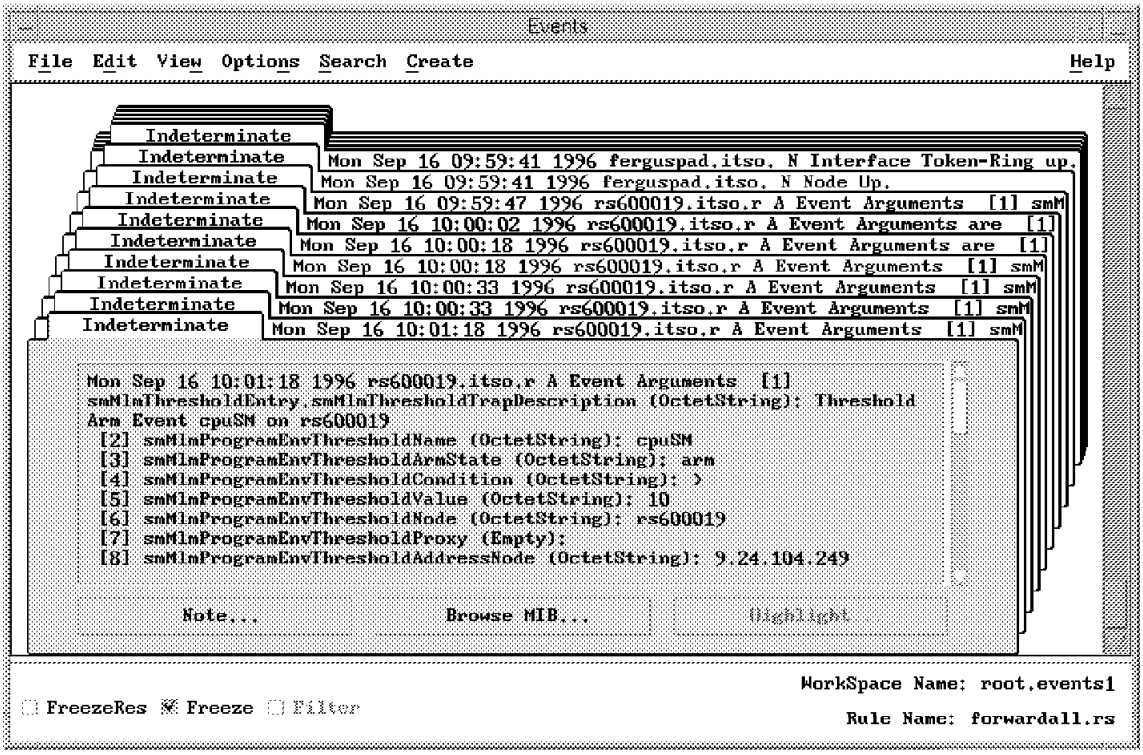


Figure 398. MLM Trap in the NetView Event Display

The two values we require to be present in the T/EC event are the node name and description. The easiest way to find the dotted decimal object IDs for these is to load the MIB file into NetView for AIX and then use the MIB browser to locate the entries. When you find the MIB variable, select it and click on **Details** to display the object ID. Figure 399 on page 417 shows the display for the trap description variable.

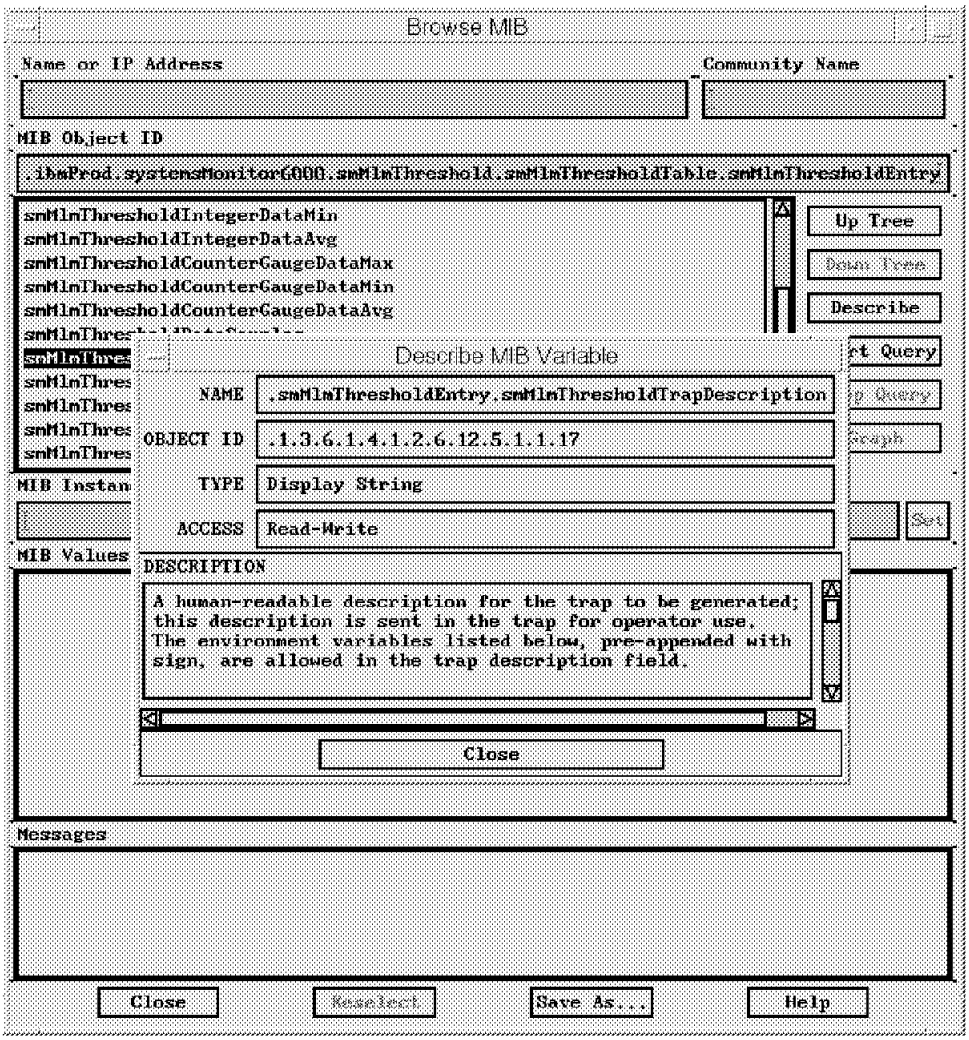


Figure 399. Using the MIB Browser to Discover Object IDs

Having extracted the object IDs in this way we updated the tecad_nv6k.oid file as shown in Figure 400.

```
#"sysmon"

"sm6000armNode"         "1.3.6.1.4.1.2.6.12.1.4.4.5.8"
"sm6000armDesc"         "1.3.6.1.4.1.2.6.12.5.1.1.17"
```

Figure 400. Extract from the tecad_nv6k.oid File

12.3.1.2 Defining a New Event Class for MLM Threshold Events

The NetView/Openview adapter assigns a different event class to each trap type that it passes to the T/EC server. We therefore had to add a new class for the MLM traps to the tecad_nv6k.baroc file. After you update this file you have to re-import the classes into the T/EC server. Figure 401 on page 418 shows the entry we added to the baroc definition.

```
TEC_CLASS :
     SM6K_ARM ISA OV_Event
     DEFINES {
     source: default = NV6K;
     };
END
```

Figure 401. Baroc Definition for Additional MLM Trap

To import this new definition, execute the following commands:

```
wdelrbclass tecad_nv6k.baroc NetView
wimprbclass tecad_nv6k.baroc NetView
wcomprules NetView
wstopesvr -d
wstartesvr
```

12.3.1.3 Defining the Mapping Between the MLM Trap and Baroc

The NetView/Openview adapter uses a CDS file, tecad_nv6k.cds, to control how trap information is mapped into event classes and slots. The entries in the CDS file are in two parts: a filter portion which is used to match the entry with a specific trap type, and the mapping portion which determines how the slot values are filled. We added the lines shown in Figure 402 to tecad_nv6k.cds.

```
CLASS SM6K_ARM
SELECT
1: ATTR(=,$ENTERPRISE) , VALUE(PREFIX, "1.3.6.1.4.1.2.6.12.5.1" ) ;
2: $SPECIFIC = 1 ;
3: ATTR(=, "sm6000ArmNode") ;
4: ATTR(=, "sm6000ArmDesc") ;

MAP
severity = WARNING ;
 hostname = $V3 ;
 msg = $V4 ;
END
```

Figure 402. MLM Additions to the CDS File

To restart the adapter with the new parameters, copy the modified CDS and OID files to the NetView server and perform the following steps:

- Copy tecad_nv6k.oid to /etc/Tivoli/tecad/etc
- Copy tecad_nv6k.cds to /etc/Tivoli/tecad/etc
- Stop and restart the NetView adapter with the following commands:

  ```
  ovstop tecad_nv6k
  ovstart tecad_nv6k
  ```

12.3.1.4 Displaying MLM Threshold Events in the T/EC Console

As usual, we added source and group definitions to allow the Systems Monitor MLM events to appear in a console display. Refer to 10.3.4, "Defining T/EC Groups and Sources" on page 336 for details of the process.

Testing the event mechanism from end to end can be performed by starting the NetView system and viewing from the TEC the actual messages being received. The events displayed in Figure 404 on page 419 will be displayed.

Important

If any section of the CDS or OID files are incorrect, the event will never be sent to the TEC server.

If nothing appears in the console display you can check that the events are being received and processed by the T/EC reception engine by using the wtdumprl command. Figure 403 shows the output of a wtdumprl command.

```
 ### EVENT ###
SM6K_ARM;source=NV6K;sub_source=NET;origin=9.24.104.249;severity=CRITICAL
 ;host=9.24.104.249msg='Threshold Arm Event from SysmoncpuSM';END
### END EVENT ###
PROCESSED' '
```

Figure 403. Output From wtdumprl

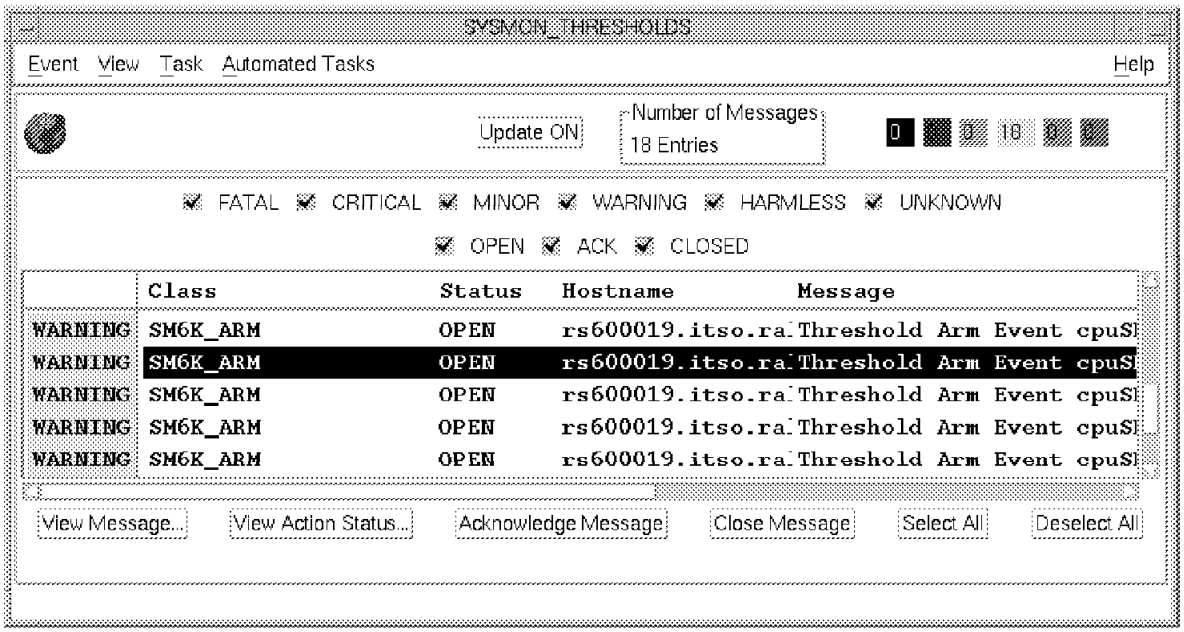


Figure 404. SM6K_ARM Events

If you click on **View Message** the event details can be seen, as shown in Figure 405 on page 420.



Figure 405. SM6K_ARM Message

12.3.2 Managing SNMP Network Devices Using the NetView Ruleset Adapter

A considerable part of the environment managed by SNMP network management stations is made up of devices such as hubs, routers and switches. To configure these traps using the NetView/Openview adapter you would have to follow the same set of steps we used for MLM traps in the last section. In contrast, we show how to add the new traps using the more current Ruleset Adapter, which is a much easier task.

To cause the 8260 events to be forwarded to T/EC we only had to modify the ruleset specified in the ruleset adapter configuration. In our case the ruleset is called TEC.rs (see 11.5, “The NetView for AIX Ruleset Event Adapter” on page 370 for a description of the adapter and the configuration process). Figure 406 on page 421 shows the final NetView ruleset. There are three *Trap Settings* nodes in the ruleset, each of which allow a certain group of traps to pass.

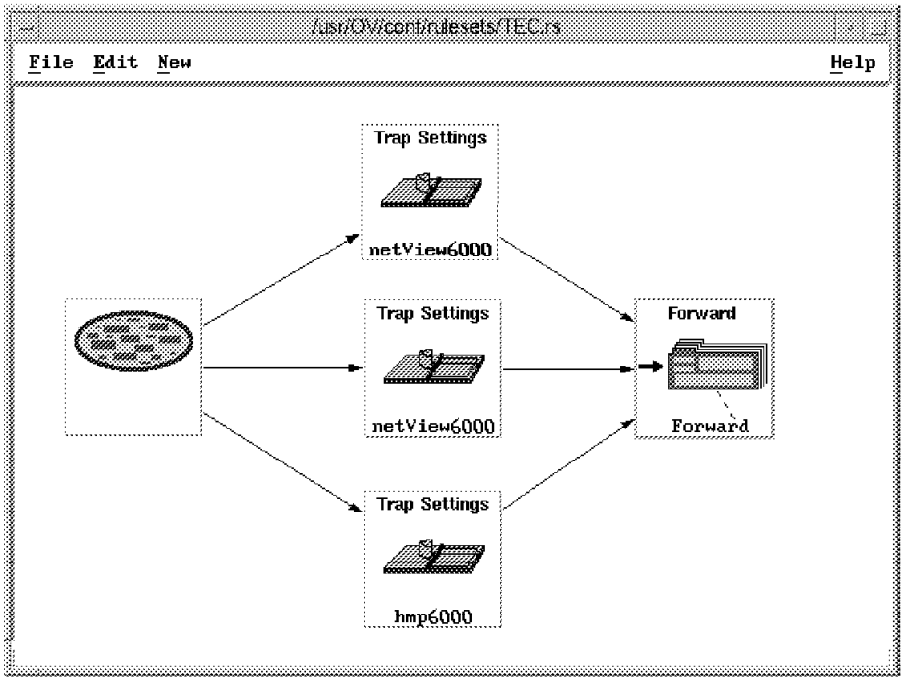


Figure 406. TEC.rs for All 8260 Alerts and NetView Node Status Events

One of the Trap Settings nodes allows *all* of the 8260 traps to be forwarded to T/EC. You can achieve this by selecting the appropriate Enterprise ID and then selecting all of the specific traps in the list. Figure 407 on page 422 shows the dialog for this.

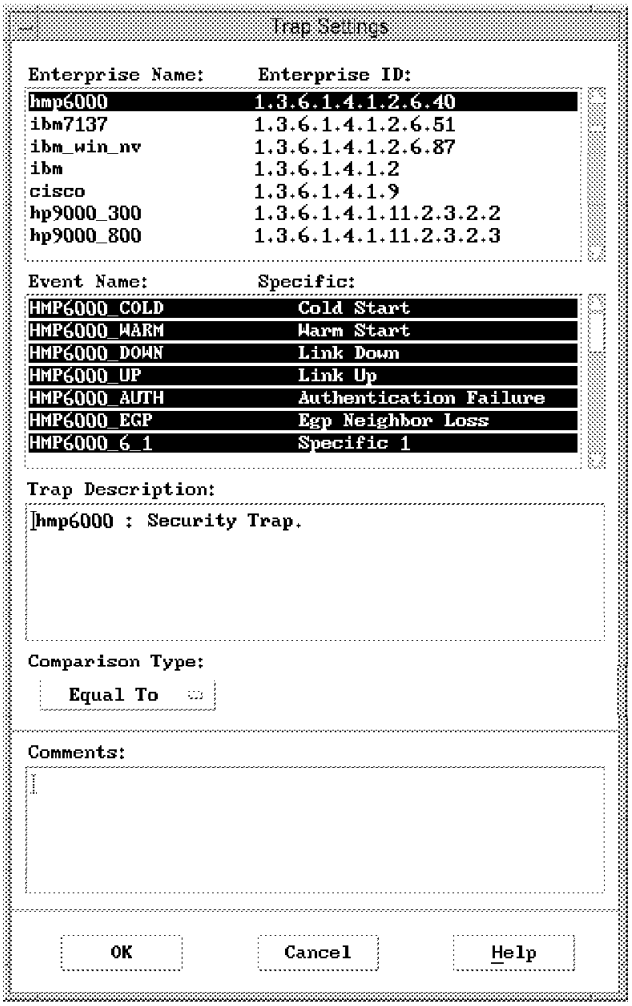


Figure 407. TEC.rs Trap Settings Definition for all 8260 Traps

We then only needed to import the nvserverd.baroc class definition and create appropriate source and group definitions to make the events appear in the console display. Figure 408 on page 423 shows a resulting event in the T/EC console.

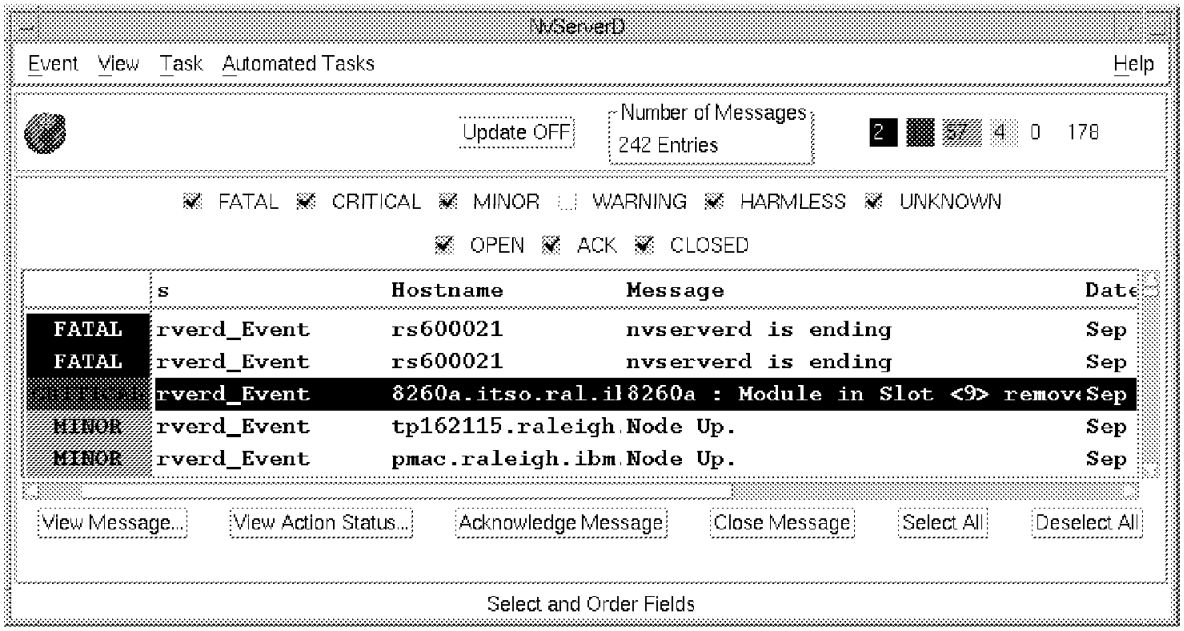


Figure 408. 8260 Event in the T/EC Display

Notice that the message text is the same as it would appear in the NetView event display. The formatted message is passed in the *msg* slot of the T/EC event. This is a very helpful feature because many traps have only basic information in them and considerable formatting is needed to make them understandable by humans.

From this simple example it is clear that the ruleset adapter is much easier to configure and extend than the older adapters for NetView and Openview. However, this convenience does come at a price: in the ruleset adapter case, every event has the same event class, so if you want to use the T/EC rules to process the events, the rule engine has to work harder to distinguish different event types from the contents of the nv_specific slot.

Part 4. Integrating Management Applications into TME

© Copyright IBM Corp. 1996

Chapter 13. Tivoli/Plus Modules

So far we have discussed the basic capabilities of the TME platform and its integrated core applications. These provide many of the functions you are likely to require when managing business processes. However, business processes are usually based on other applications, toolkits and development environments. Each of these environments poses its own management challenges, for example: installation, customization, configuration and monitoring. Ideally, someone will have already addressed these challenges using TME, so you do not need to re-invent the solutions. That is exactly what a Tivoli/Plus module is.

Tivoli/Plus modules provide the function to integrate applications into the Tivoli Management Environment. Generally they do not use the *tight* integration possibilities of TME. That is to say, they do not create new profile types or interface directly with the object request brokers. Instead, they are more often collections of pre-defined configurations and scripts that customize the TME desktop and core applications to support a particular environment.

In this chapter we show some examples of existing Tivoli/Plus modules.

13.1 Tivoli/Plus Modules in Practice

Normally, the modules install desktop resources under the Tivoli/Plus collection icon shown in Figure 409.

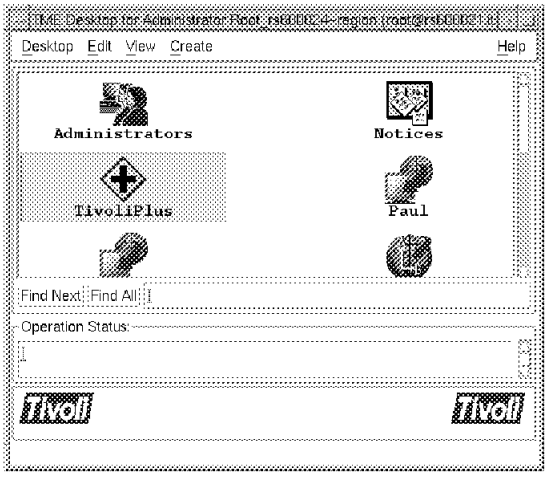


Figure 409. The Tivoli/Plus Icon

Each plus module adds a subdirectory under /usr/local/Tivoli/bin/generic_unix/TME/PLUS. These subdirectories hold the executables and configuration files relating to the specific Tivoli/Plus module. This directory structure will be created on both the server and client machines.

A typical Plus module comprises one or more of the following:

- TME tasks and jobs
- Predefined Sentry monitors
- Profile managers
- Predefined Courier distribution packages

© Copyright IBM Corp. 1996

- T/EC server integration (including event classes and, sometimes, rules)

The Tivoli/Plus modules take advantage of the existing applications provided by TME 10 (such as TME 10 Sentry, TME 10 Courier, and TME 10 Enterprise Console). This provides a common interface to all applications and eases the management issues and problems that occur when trying to manage a diverse, distributed network.

Each /Plus module provides its own documentation. In preparing this chapter we used the following manuals for reference:

Tivoli/Plus User's Guide,

Tivoli/Plus Release Notes,

Tivoli/Plus for AR System,

Tivoli/Plus for ADSM,

We illustrate the /Plus module concept using two examples:

- Remedy Action Request System Tivoli/Plus module
- ADSM Tivoli/Plus module

As always, before installing any new TME 10 products we recommend that you perform a database backup.

13.1.1 The Remedy Action Request System

Tivoli/Plus Module

ARS is a solution from Remedy, a member of the Ten /Plus partners association, for help desk, problem management and other work flow-based systems management functions. This section deals with the Tivoli/Plus module only and does not cover the installation and customization of the Tivoli Action Request software. You should refer to Remedy documentation for this.

The ARS /Plus module contains a number of elements:

- TME tasks and jobs to perform regular maintenance activities
- Pre-defined file packages to install ARS client code
- Sentry monitors for client and server systems
- Trouble Ticket integration with the TME 10 Enterprise Console

We do not examine all of the features, but concentrate on the latter one.

13.1.2 Installation

At the time of writing, the ARS /Plus module was only available as a TME 2.5 product. TME 2.5 required you to provide a license key when installing any product. This meant that we had to load the application from the command line, because the GUI dialog does not support license keys in TME 3.0. The command is as follows:

```
winstall -c /:/tivoli3cd/Tivoli_Plus_REVE -i REMEDY_A.IND
-s rs600024 -l <license key> rs600024
```

After the product has been installed there is a new icon under the TivoliPlus icon on the TME Desktop (see Figure 410 on page 429).

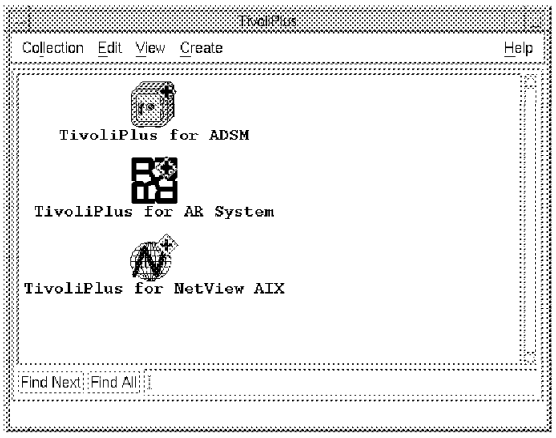


Figure 410. Remedy Plus Module Icon

We also installed the AR application on the T/EC server, rs600020, and the AR application server, rs600013. This machine must also be a managed node. The AR Plus module uses the Sentry application to monitor the AR server daemons and log files. This requires that Sentry is also installed on the server (rs600013 in our case) for the monitors to function.

The first post-installation task we performed was to set up the T/EC environment to allow the T/EC to receive events from ARS. To do this we double-clicked on the **Setup Event Server for ARS** icon shown in Figure 411.

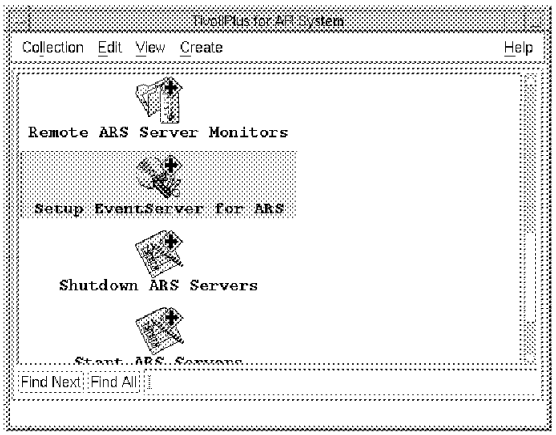


Figure 411. Remedy Plus T/EC Installation

We entered our rulebase name of Tivoli. The installation dialog will import the ARS classes and rules into this rulebase, we could have entered either a new or existing rule base.

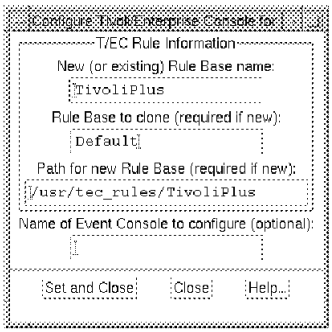


Figure 412. Remedy Plus T/EC Installation

The output from installing the T/EC components is shown in Figure 413.

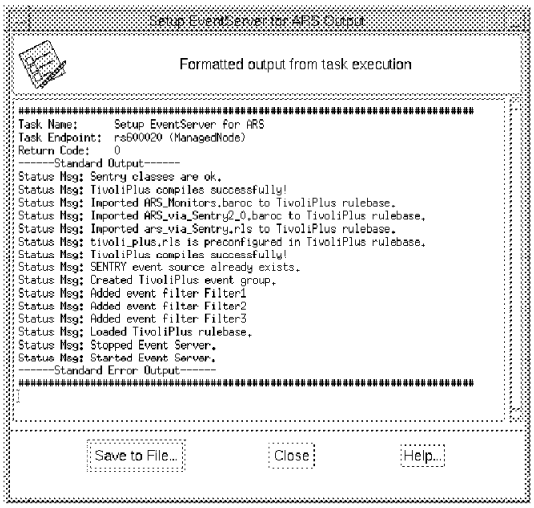


Figure 413. T/EC Install Messages

The task also updates the T/EC source and group associations, see Figure 414.

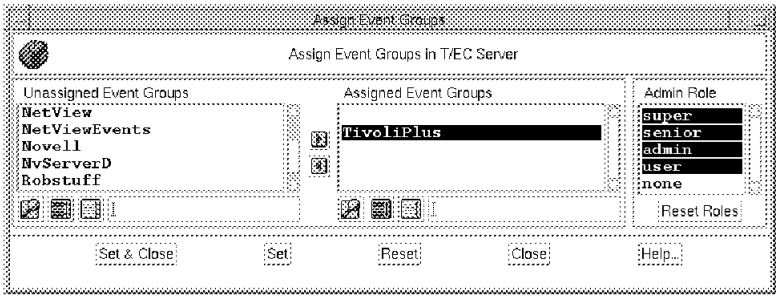


Figure 414. T/EC Group Configuration

The Remedy server machine in our lab environment was rs600013. This machine was installed as a managed node. The application scripts used by the /Plus module are stored in directory /usr/local/Tivoli/bin/generic_unix/TME/PLUS/ARS.

The /Plus module expects all ARS servers to be subscribed to a special profile manager named *ARS Server List*. This allows the components of the module, such as file packages, monitors, and jobs, to be fully pre-defined, because they can all define ARS Server List as a subscriber. To enable the module you

simply have to add your ARS server hostnames to the ARS Server List by double-clicking on the **Server** icon shown in Figure 415 on page 431.

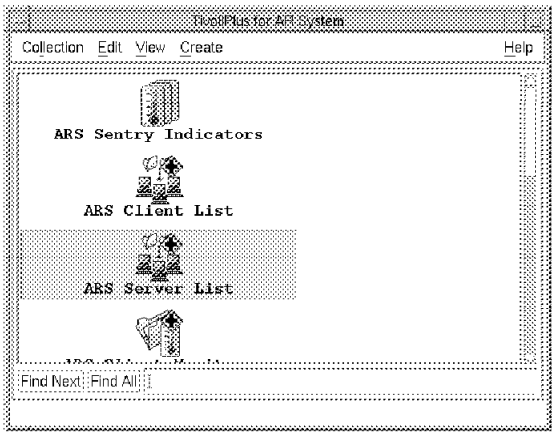


Figure 415. Add Remedy Server

We added rs600013 to this list as shown in Figure 416.

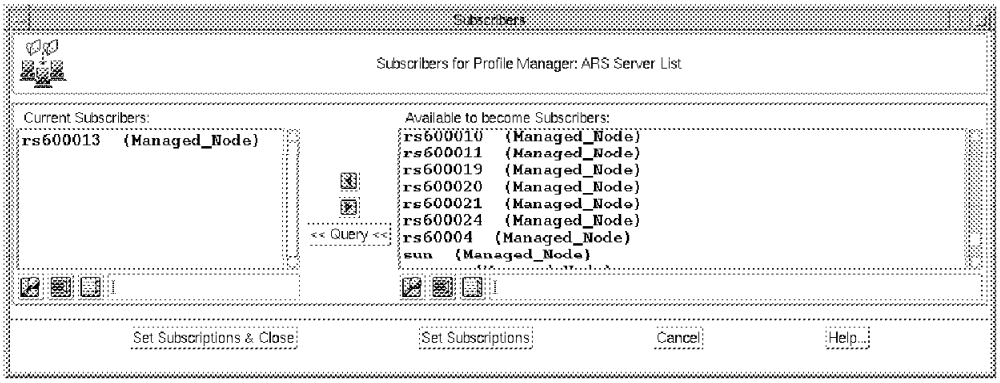


Figure 416. Subscriber Information for ARS Servers

There is an equivalent profile manager, ARS Client List, that acts as a container for the client systems. Once the two lists are created, all of the functions provided by the module, as shown in Figure 417 on page 432, may be invoked.

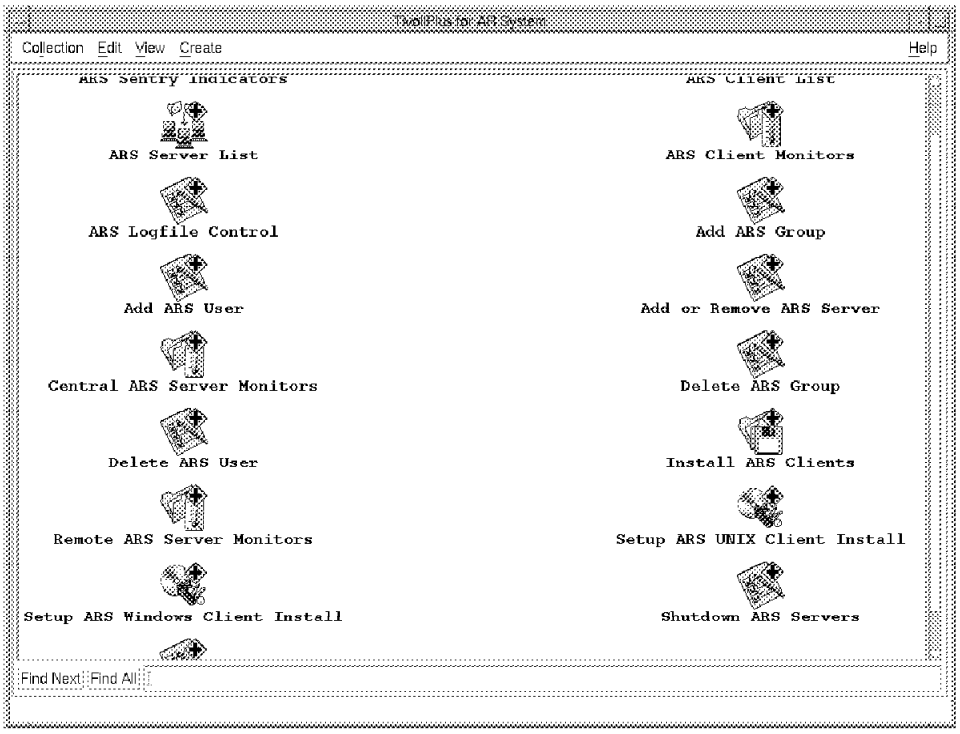


Figure 417. ARS Collection

13.1.3 Using the ARS /Plus Module T/EC Integration Feature

The T/EC console can be linked to the AR application to generate trouble tickets from events received at the console. The mechanism for doing this is a standard part of T/EC. It is implemented by a script called TroubleTicket.sh, placed in directory $BINDIR/TME/TEC. Before the function can be invoked a number of further configuration steps are needed:

- The Tivoli schema must be imported into the Remedy database.
- The T/EC server has to be a Remedy client.
- The Tivoli macro required by the Remedy software must be installed on the ARS server.

The troubleticket.sh script also has to be modified to reflect your particular environment. In our case we updated the script as shown in Figure 418.

```
aruser -d $INST_DIR/generic_unix/TME/PLUS/ARS -e Tivoli-TT.macro
-p "User=Demo" \
-p "Server=rs600013"
```

Figure 418. Customized TroubleTicket.sh Script

Now an ARS trouble ticket can be created by selecting an event in the T/EC display, then selecting **Trouble Ticket** from the menu bar as shown in Figure 419 on page 433.

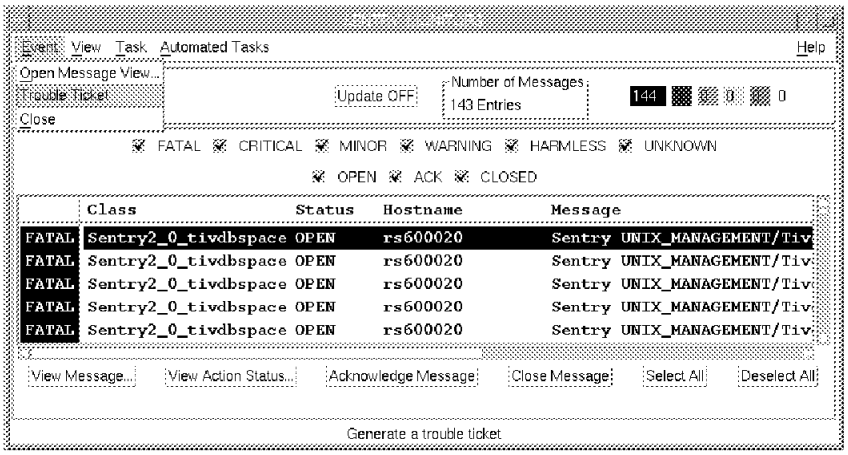


Figure 419. Creating a New Trouble Ticket

There is no confirmation at his point that the record has been created. But when we viewed the records from the AR user interface we could see the event as a database record. Figure 420 shows such an event.

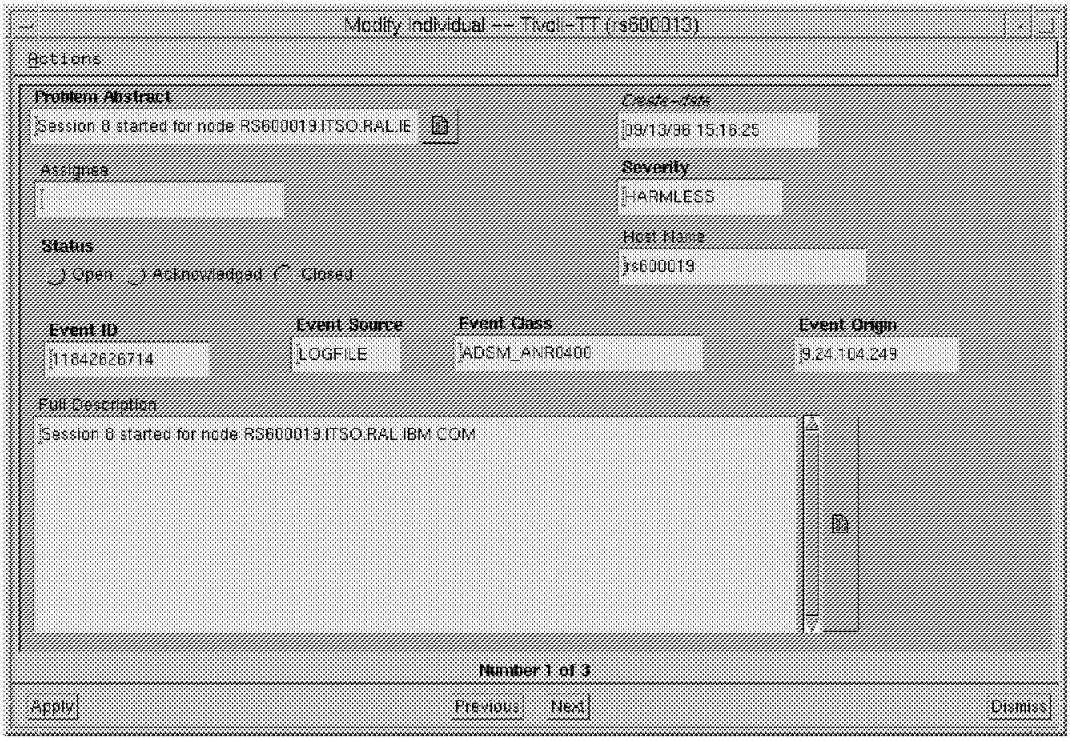


Figure 420. The Detailed Description of the Event

Figure 420 displays the detailed information that the AR system has been passed. When the trouble ticket is closed, the corresponding T/EC event is also automatically closed. Within ARS this is implemented by a call to the CloseTivoliTicket script.

It is very likely that you would want to modify this behavior to fit your particular administrative work flow. For example, you may want to close the T/EC event as soon as the ticket is successfully created in ARS, rather than when it is subsequently closed. The behavior can be easily modified, since it is all implemented by ARS filter actions. Figure 421 on page 434 and Figure 422 on

page 434 show the list of filter actions, and the detail of the filter that is invoked when the trouble ticket is closed or acknowledged. You can see the invocation of CloseTivoliTicket in this filter. To customize the behavior you could simply modify the filter conditions so that it is invoked at some other point in the ticket life-cycle.

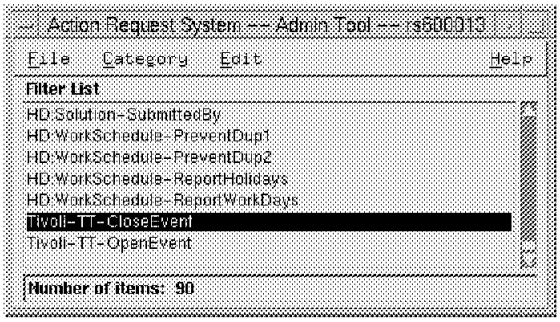


Figure 421. List of ARS Filter Actions



Figure 422. Editing an ARS Filter Action

13.2 ADSM Tivoli/Plus Module

ADSM is a comprehensive solution for data backup and archive in a distributed environment. Like many such applications it provides a powerful GUI for configuring the ADSM server and backup policies. The ADSM /Plus module does not attempt to replace that GUI, but enhances it by leveraging the capabilities of the Tivoli Management Environment.

The module contains a number of components:

- Pre-defined Sentry monitors for ADSM server functions
- Tasks and jobs for executing maintenance functions across multiple ADSM servers
- Pre-defined file packages for installation of servers and ADSM backup clients
- Extensions to the T/EC log file adapter to handle ADSM console messages

We set up a simple sample environment, installing the following software components on an AIX system, rs600019:

- The ADSM/6000 Server
- TME managed node
- TME 10 Remote Monitoring (Sentry)
- The T/EC log file adapter

13.2.1 Installation

The ADSM/6000 software is loaded in the same way as any other TME product, as described in 5.2, “Installation” on page 109. In addition to our server machine we set up a number of client machines:

- RS6000
- OS/2
- Microsoft Windows NT
- HP/UX

Because the /Plus module makes use of TME 10 core applications, the pre-installation tasks for the ADSM module include installing and configuring the following products:

- Each machine must be either a TME server or TME managed node
- TME 10 Software Distribution (Courier)
- TME 10 Remote Monitoring (Sentry)

Once we had installed the ADSM /Plus module we discovered that there were two ways in which the ADSM Server could be started. The standard ADSM/6000 installation adds a line to the file /etc/inittab. Alternatively, the /Plus module installs a script to start the server. Both these startup scripts start the ADSM Server program called dsmserv. The main difference is that the TPLUS module re-directs the output to a log file (by default, /etc/Tivoli/adsm.log). The module also updates the T/EC log file adapter to monitor this file for error messages.

We chose to amend the /etc/inittab entry to call the TPLUS startup script. We modified the adsm starup line to read:

```
/usr/local/Tivoli/bin/generic_unix/TME/PLUS/ADSM
```

There may also be a requirment to include the initiation of the log file adapter process either in the inittab file or an /etc/rc file. If the log file adapter is not started no messages will be sent to the TEC server. We also had to modify the startup script to set the language correctly. The $LANG variable had to be set to En_US. This is because we were running the adapter on an AIX 4.1 system, which was not officially supported at the time of writing. Full support for AIX 4.1 has since become available.

The installation process performs a number of actions, including the following:

- It modifies the tecad_logfile.cds and tecad_logfile.fmt files. These files are modified to include the ADSM class definition.
- It modifies the tecad_logfile.conf to include the source file /var/spool /Tivoli/adsm.log. This append directs the log file adapter to look for messages in the adsm.log file in addition to the other defined sources.
- It adds a new collection to the Tivoli database for ADSM. This is located below the Tivoli/Plus Icon. If this collection does not exist the installation process will create it (see Figure 423).

You can display icons representing all the facilities of the ADSM /Plus module by double-clicking on the new icon.

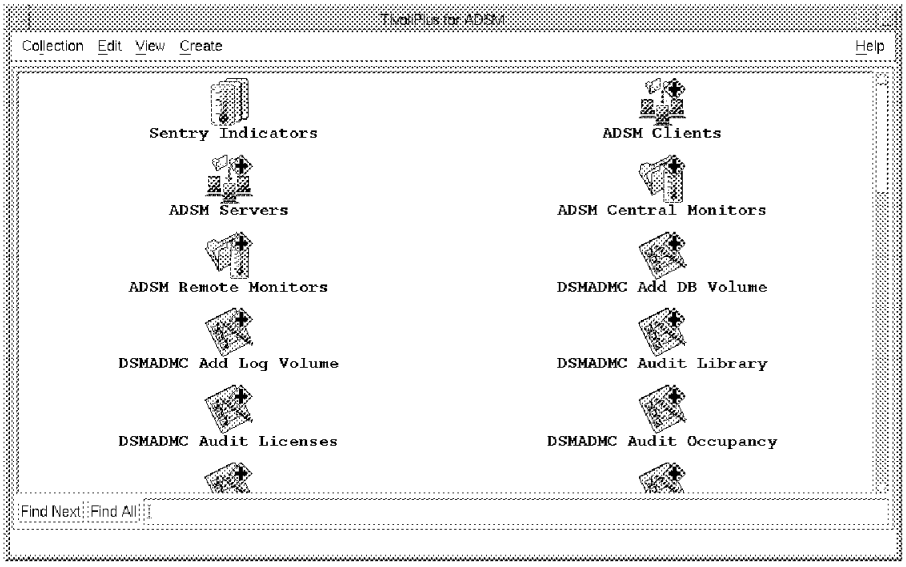


Figure 423. The ADSM Collection

13.2.2 Using the ADSM /Plus Module to Install ADSM Backup Clients

One of the elements of the /Plus module is a set of pre-defined installation packages for ADSM clients. This is a good example of using a combination of TME functions to simplify system configuration.

Before doing any software distribution we had to make the agent code available, by installing it in a directory on a managed node (in this case, on the TME server). Then we configured an OS/2 agent as follows.

From the Tivoli/Plus for ADSM collection click on the **Setup ADSM OS/2 Client Distribution** icon. This is a dialog that creates the operating system specific information for a file package, without having to go through the full file package

creation process (refer to Chapter 5, “Tivoli/Courier” on page 103 for details of this). The panel for OS/2 is shown in Figure 424 on page 437.

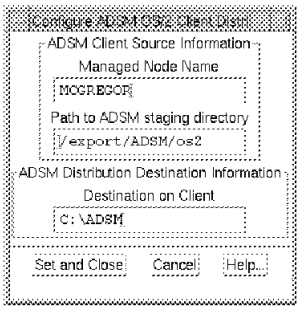


Figure 424. Defining Package Details for OS/2

There are similar options for other operating systems. In the case of UNIX, the ADSM client code is different for each system type, so there are separate options to set up each one. Figure 425 shows one of the AIX options.

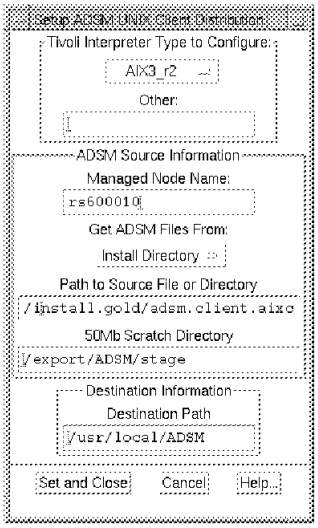


Figure 425. Defining Package Details for an RS6000 Client

In both of these cases the Managed Node Name is where the source file is stored, not the target for the software distribution.

The result of these definitions is that new courier file packages are defined and placed under the Distribute ADSM Software icon in the ADSM collection. Figure 426 on page 438 shows this icon and Figure 427 on page 438 shows the file packages that were created for OS/2, and two versions of AIX.

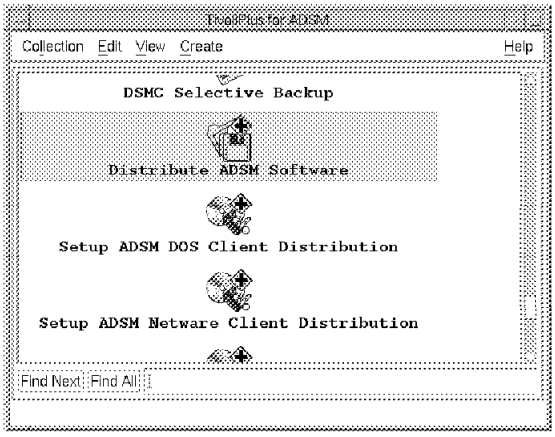


Figure 426. Distribute Client Software Icon

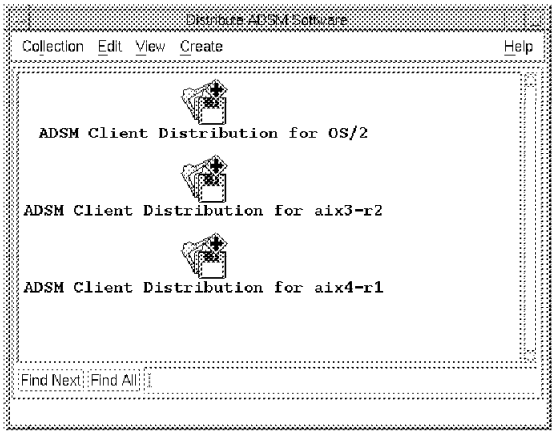


Figure 427. Generated File Packages

The file packages can now be distributed to the client nodes. The ADSM /Plus module implements a profile manager hierarchy, with separate profile managers for each client system type which are all subscribed to a single profile manager named ADSM clients. To distribute the client software we subscribed the new client systems to the appropriate profile managers.

At this stage we decided to send all task output errors to the TEC. This requires creating and configuring a log file adapter. The default output will be sent to the notice group *courier*. This involves creating a new event source for the log file adapter. To achieve this we followed the steps below.

This process can be adopted and customized for any Tivoli tasks running in the Tivoli framework. The basic steps are to create a new source for the log file adapter, then creating a new cds and fmt file for the new messages. On the TEC server a new baroc file must be created using the class definitions found in the cds file. Finally a new TEC group is defined and linked to the required console.

The output of a task was monitored to see if any patterns where found so that the adapter could filter them. For this we set the Output to File to 0n and set the file as /tmp/adsm_task.output (see Figure 428 on page 439).

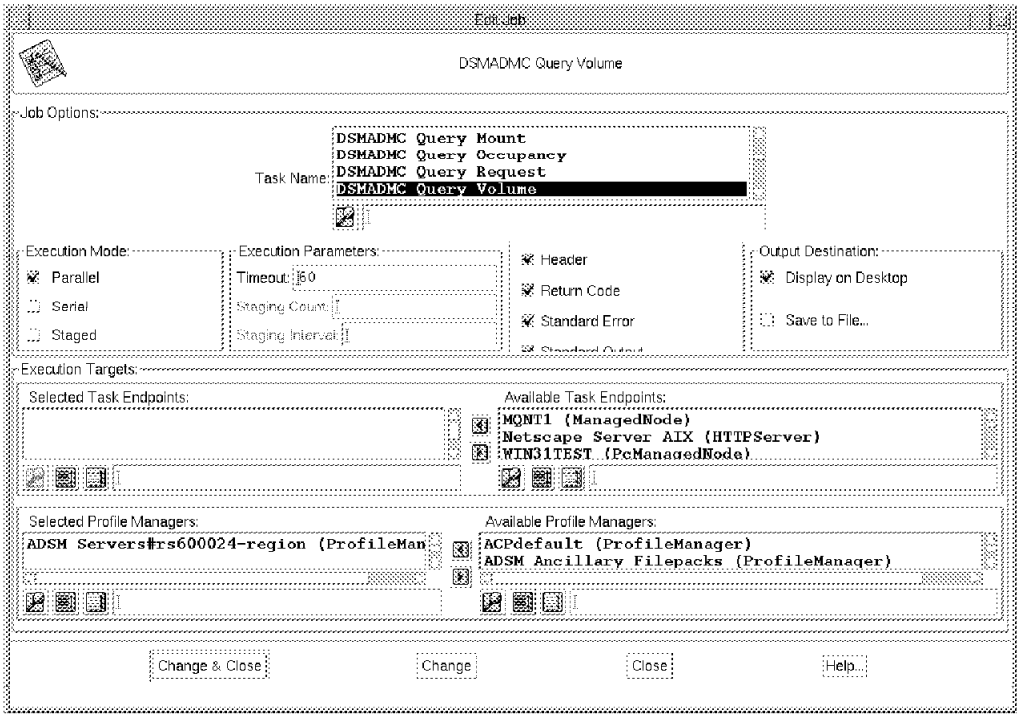


Figure 428. File Package Definition with Log File Option Enabled

13.2.3 ADSM Log Interface with the TME 10 Enterprise Console

The ADSM /Plus module provides a facility to pass ADSM log messages to the T/EC. This is an extension to the normal log file event adapter. Refer to 12.2, "Extending the Logfile Adapter to Manage a Distributed Application" on page 393 to gain an understanding of the steps that the module takes to do this.

The ADSM /Plus module provides a TME task to configure the components required for the TEC console installation, including updating the following files:

tecad_logfile.cds This file is located in the directory /etc/Tivoli/tecad/etc and resides on the host where the log file adapter is installed, that is, the ADSM server. It provides a mapping between the log messages and baroc classes.

tecad_logfile.baroc This file contains the class definitions for ADSM.

We configured the T/EC extensions by double-clicking the **Setup EventServer for ADSM** icon in the ADSM /Plus collection. After doing so there was a new T/EC event group defined. Figure 429 on page 440 shows the event group icon and Figure 430 on page 440 shows some sample messages from the ADSM server.

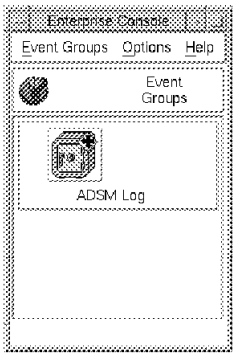


Figure 429. T/EC Event Console Group Definition

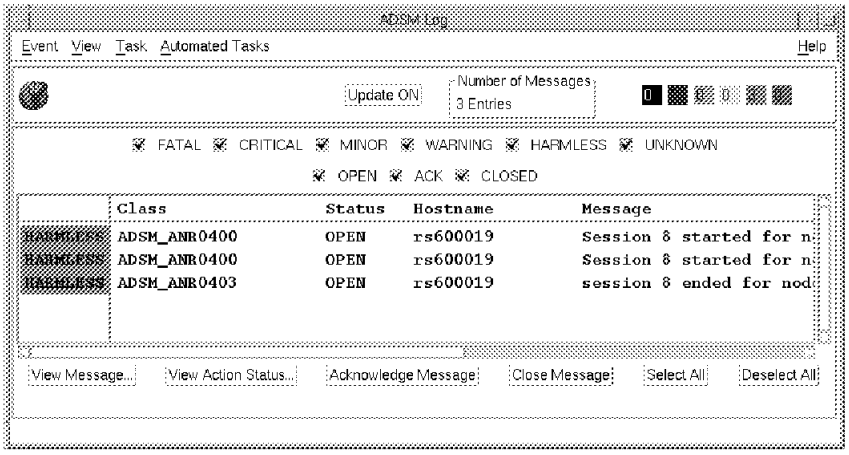


Figure 430. T/EC Events Created from the ADSM Module

Chapter 14. TME 10 Net.Commander

Many organizations are currently providing information using the Internet. Many more are planning to do so, or are enhancing their existing presence. Internet technology is also being exploited within corporate networks, in the form of intranets. These exciting developments involve the installation of Web servers, mail servers, browsers, firewalls, etc. A number of issues emerge from this, such as how to keep the servers running all the time, and how to minimize the cost of adding new systems to the network; in fact, all of the questions that accompany any new distributed system rollout. TME 10 Net.Commander is a ready-built application for deploying and monitoring these systems.

In many ways, Net.Commander is more akin to a /Plus module than a core TME application. It does create some new profiles within the TME object database, but it is primarily a collection of tasks, monitors, file packages and event definitions for use with the other TME core applications. TME 10 Software Distribution (Courier), TME 10 Remote Monitoring (Sentry) and TME 10 Enterprise Console are therefore pre-requisites of Net.Commander.

14.1 Installing Net.Commander

The installation of the Net.Commander is similar to other Tivoli component installations, as described in 5.2, “Installation” on page 109. It needs to be installed on the TME server and also on the Internet application servers. We have already mentioned that a number of other TME 10 products need to be installed first. Table 15 lists the prerequisites.

Table 15. Components Required by Net.Commander on Different Servers

Server type	**Preinstalled Components**			
	TME Platform (incl. TASKS)	**Sentry Monitor**	**T/EC Log File Adapter**	**Courier**
WWW Server	X	X	X	
News Server	X	X		
Mail Server	X	X		
Firewall			X	
Browser Distr.	X			X

We tested Net.Commander in a simple environment comprising only a manager machine and an Internet server. The manager machine in this case is also the TME server, but that is not essential. The hostname of the Internet server in our example was rs600021 and of the manager node, rs600021.

After you have installed the product you will see a new icon on your administrator desktop, as shown in Figure 431 on page 442.

© Copyright IBM Corp. 1996

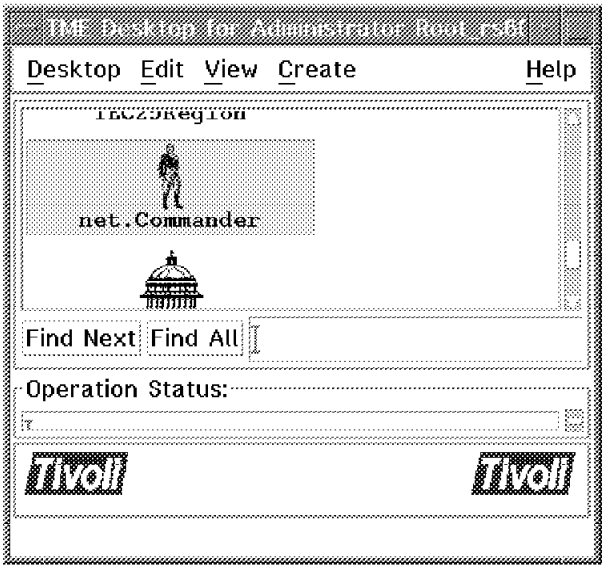


Figure 431. Net.Commander Collection Icon

This icon represents a policy region, within which you can find all of the Net.Commander profile managers, tasks, and sentry monitors. If you double-click the icon you will see the contents of the policy region as shown in Figure 432.

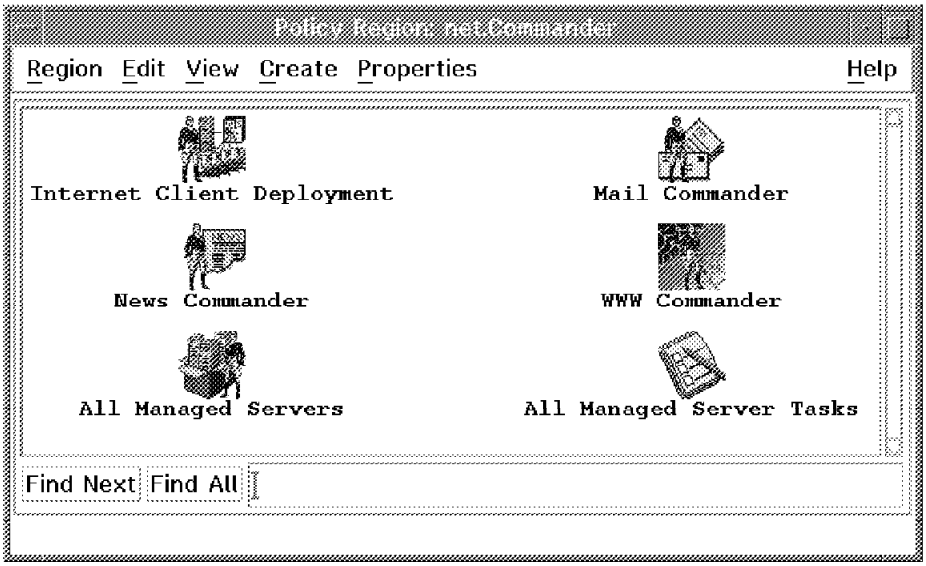


Figure 432. Net.Commander Policy Region Contents. Note that we were working with a pre-release version of Net.Commander. The contents, appearance and behavior of the final version may not be identical.

14.2 Customizing and Managing a Web Server

Currently, Net.Commander will manage the installation and operation of the following Web servers:

- NCSA (UNIX only)
- Apache (UNIX only)

- Netscape (AIX, Sun, HP, Windows 95, Windows NT, Windows 3.x)

In our case we elected to use Netscape Enterprise Server 2.0 on an AIX platform.

We installed the server manually and then configured TME 10 Net.Commander as follows. Double-click on the **WWW Commander** policy region icon and select **Create** followed by **HTTP Server...** from the menu bar (see Figure 433).



Figure 433. Create a New HTTP Server

Fill in the requested data and click on **Create & Close**. Make sure that all directories and the name of the node are correct. You can also specify a log archive directory and one or more web master mail alias(es). Figure 434 on page 444 shows the panel for creating a new Web server.

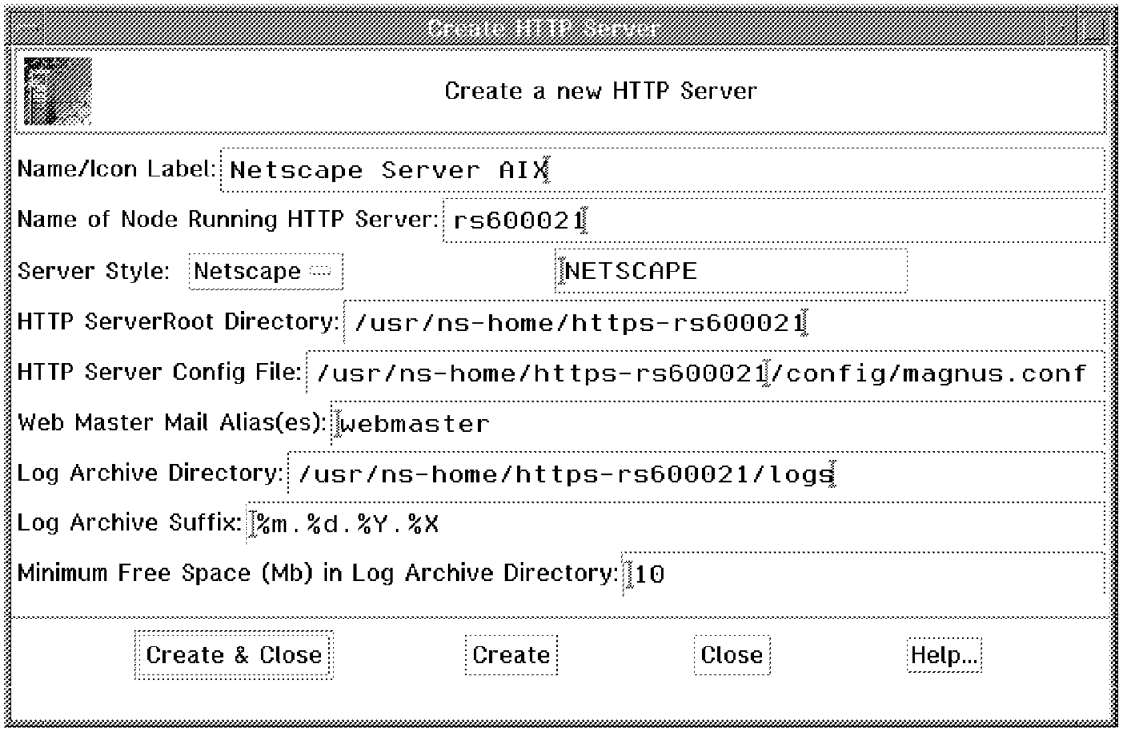


Figure 434. Create an HTTP Server

A new icon now appears in the WWW Commander policy region. From the context menu of the icon you are able to stop and start the server. You can now manage the server using Net.Commander tasks and monitors.

To manage the server it must be defined as a subscriber to some profiles. Net.Commander uses a hierarchy of profile managers to simplify the subscription process. There is an All Managed Servers profile manager which has three other profile managers subscribed to it: WWW servers, mail servers and news servers. It is a little bit different for the firewalls, as we discuss below. So to add the new server into the hierarchy you need to subscribe it to the WWW Servers profile manager. You do this by selecting the profile manager icon with the right mouse button and clicking on **Subscribers...** (see Figure 435 on page 445).

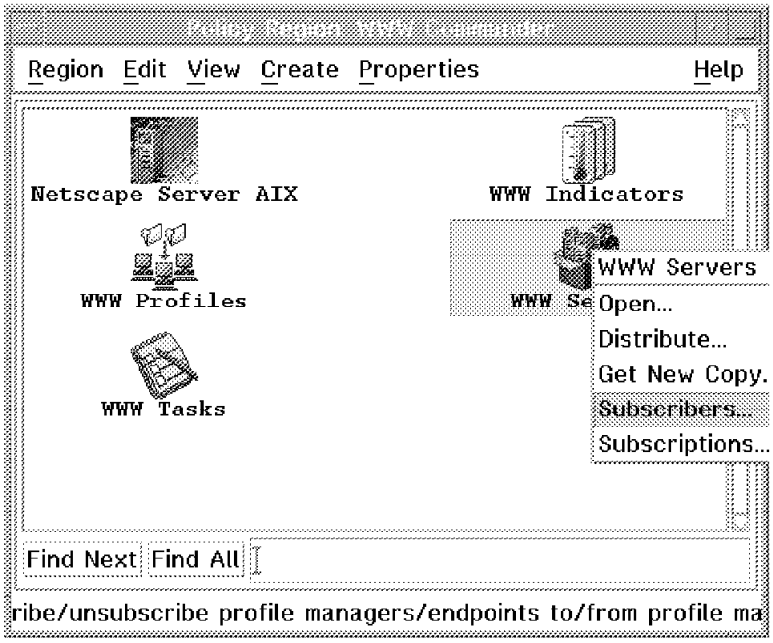


Figure 435. Adding a Subscriber to the WWW Servers Profile Manager

Add the new server (Netscape Server AIX in our example) to the Current Subscribers list by double-clicking or selecting the arrows, then select **Set Subscriptions & Close**.

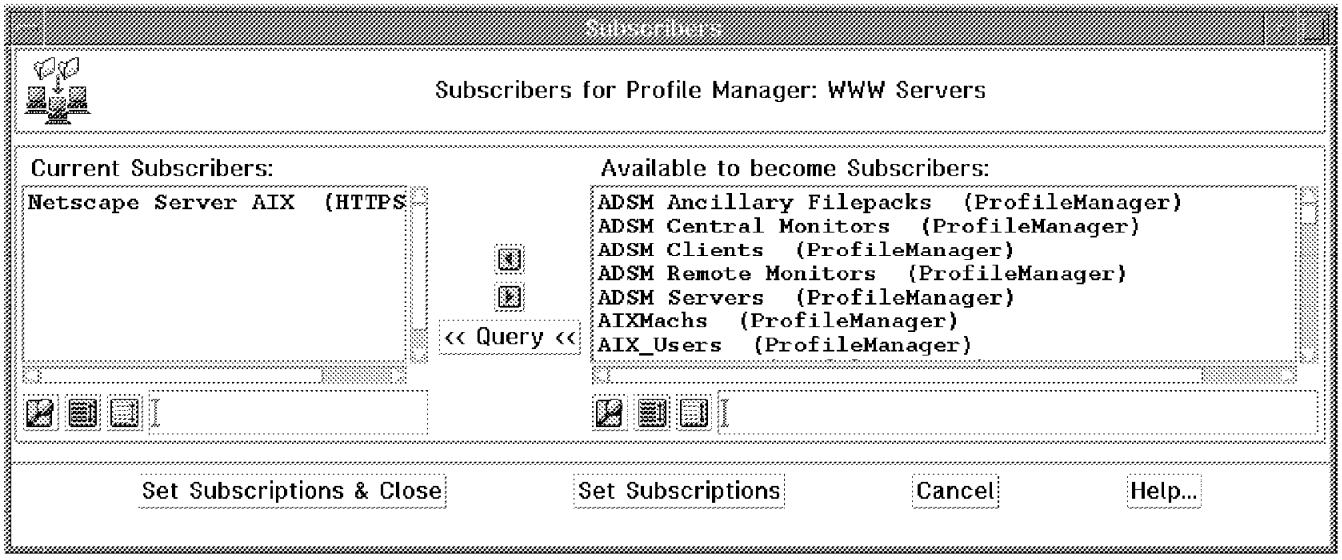


Figure 436. Subscribers for Profile Manager: WWW Servers

14.3 Using Net.Commander Functions

In this section we briefly describe the facilities that Net.Commander provides.

14.3.1 Tasks and Jobs

There are a lot of predefined tasks and jobs to handle the server. If you have more than one server, you can handle all of them with a single action instead of having to work with each server on its own.

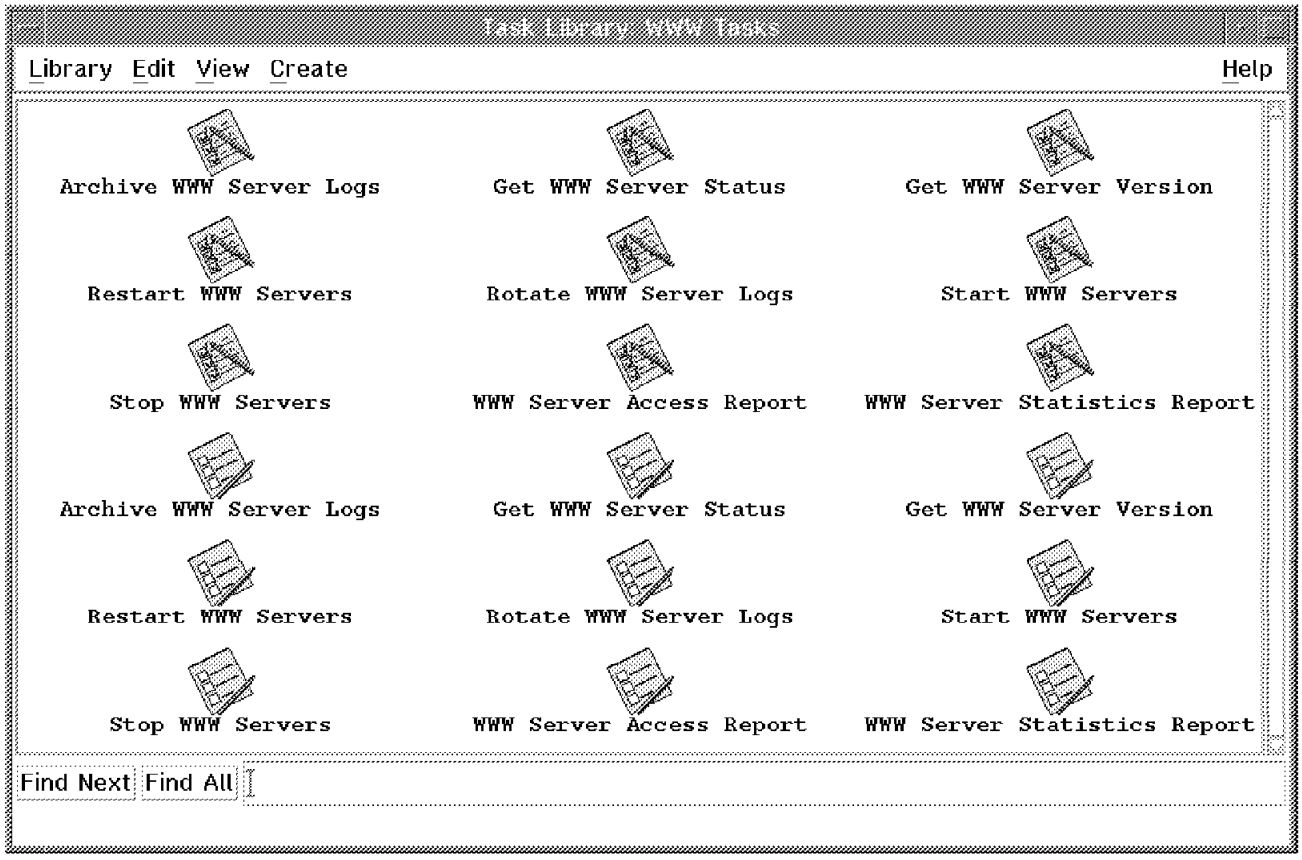


Figure 437. Tasks and Jobs for WWW Servers

In addition to tasks for starting and stopping the Web servers, there are tasks to get information from the Web server reports. This is very useful to get consolidated usage statistics without using the administration tools on the web server itself.

For a complete description of the tasks refer to the *Tivoli/Net.Commander User's Guide*.

14.3.2 Monitors for the Web Server

A new Sentry monitoring collection will be added by the installation. From this collection you can add predefined monitors to the WWW Monitors profile manager. There are monitors for file sizes, process status, log files and transmitted data. Figure 438 on page 447 shows the standard monitoring schedule for a number of Web server monitors.

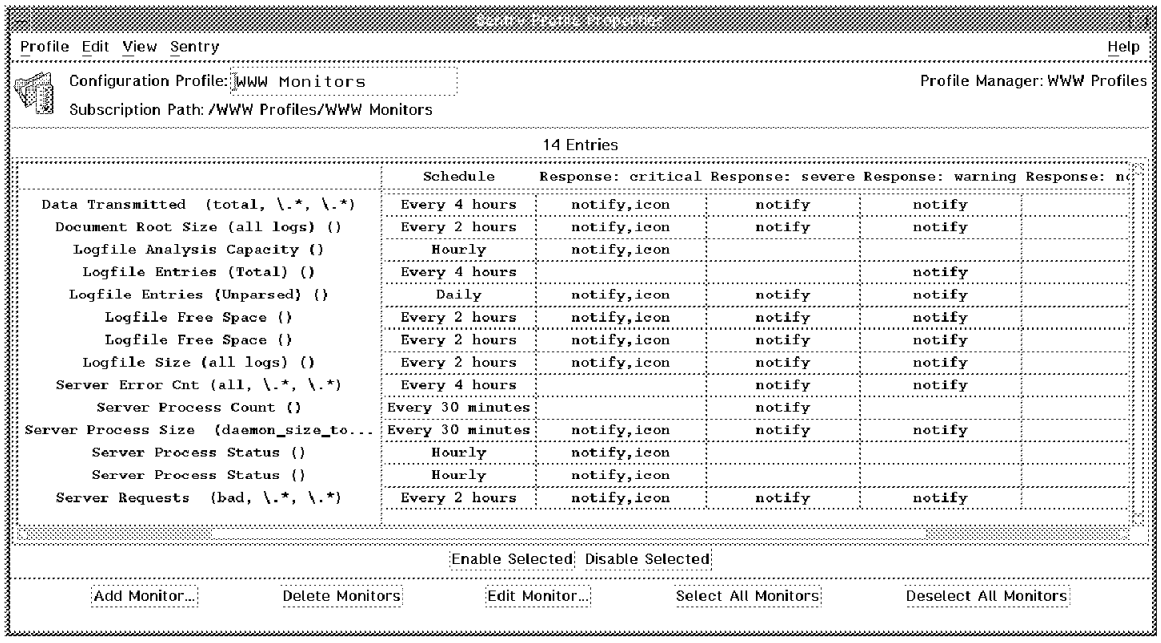


Figure 438. Sentry Monitoring Schedule

You have to add each monitor in one step. But you also can add monitors from the command line. The default monitors are added by a shell script:

```
/usr/local/Tivoli/bin/generic/TME/NETCMDR/WWWSVR/Http.sh
```

You can change the shell script to add all monitors with default values to your WWW Monitors profile by changing the following line:

```
SP="TivoliSentryDefaults#$IROname"
```

to

```
SP="WWW Monitors"
```

In addition to this, you have to change all occurrences of the string $SP to "$SP". Now you can run the shell script to add the monitors to the WWW Monitors profile.

As a last step for the installation procedure of the monitor, distribute the profile to the WWW Servers profile manager (which contains all of the Web servers that you support).

14.3.3 Getting T/EC Events from the Web Server

Web server errors, such as failed logins to secure directories, forbidden access or files not found can be circumstantial evidence of cracker activity, or may be evidence that the document content needs maintenance. With Net.Commander it is possible to send these errors as events to the TME 10 Enterprise Console server and display them on an event console. Once the events are part of the T/EC event stream you can use rule processing to correlate and filter them.

There are a few steps required to install this feature:

1. Create a file: /etc/Tivoli/tec_server. Within this file, there must be only the name or IP address of the event server. In our example, this is rs600020.itso.ral.ibm.com.

2. Import the new Net.Commander T/EC classes into the current rule base. The procedure for importing baroc classes is explained in 10.3.3, “Importing Event Classes” on page 332.
3. Compile the rule base, using the GUI or the wcomprules command.
4. Reload the T/EC rule base using the GUI or the wloadrb command
5. Stop and restart the event server, using the GUI or the wstopesvr/wstartesvr commands.
6. Customizing the Netscape Enterprise Server:

 You first have to access the Netscape Commerce Server Manager dialogs. You do this using a normal Web browser. Select the URL of your enterprise server machine, without specifying a file name. Then select the **Error Responses** option from the **Systems Settings** menu. You will see the panel shown in Figure 439.

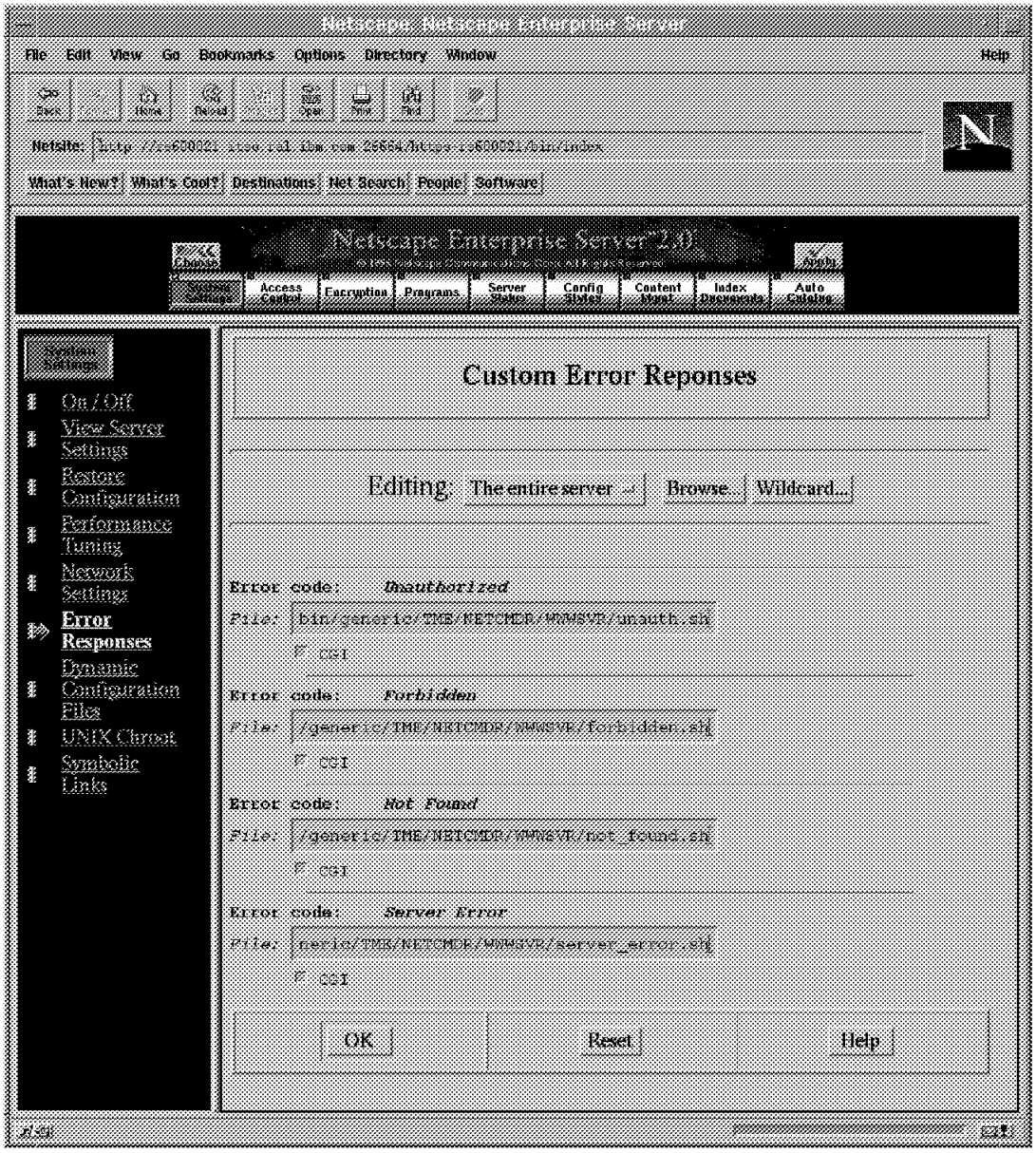


Figure 439. Error Response Settings in Netscape

Associate the following file names with the appropriate error event. Use the whole path name:

Unauthorized
/usr/local/Tivoli/bin/generic/TME/NETCMDR/WWWSVR/unauth.sh

Forbidden
/usr/local/Tivoli/bin/generic/TME/NETCMDR/WWWSVR/forbidden.sh

Not Found
/usr/local/Tivoli/bin/generic/TME/NETCMDR/WWWSVR/not_found.sh

Server Error
/usr/local/Tivoli/bin/generic/TME/NETCMDR/WWWSVR/server_error.sh

Identify these as CGI scripts. The scripts will be executed by the Netscape server. This is important, because the server normally runs under the nobody user ID. This is not a problem, except that if you want to execute the programs or the postemsg command manually, you should also start them under user ID nobody. The reason for this is that the first time you execute the scripts, a file (/tmp/postemsg.out) will be created owned by your current user ID. Under UNIX, files in /tmp receive special protection, so they cannot be updated by another user Id. Unfortunately you will not see any error messages while the script is executed by the Netscape server. The only way is to redirect the output of the command to the Web browser or to another file. For testing this, it is possible to redirect to the browser:

```
echo `postemsg  -S "$TEC_SERVER" -r WARNING ...`
```

This works because the script is run by the CGI interface. But, you should not do this in your production environment, because everybody will see a message. It is better to redirect the output to a file:

```
echo `postemsg  -S "$TEC_SERVER" -r WARNING ...` >> /tmp/WWW_TEC.log
```

7. Test the Netscape server events.

 Start the Web server and test the T/EC connection by trying to access a Web page that does not exist. An event should be sent to the T/EC. Note that these are log file events. If you have not already added the log file event source into an event group filter, you will not see them on the T/EC console (refer to 10.3.4, “Defining T/EC Groups and Sources” on page 336 for an explanation of this). However you can check that the events are being received and recognized by the event server using the following command:

   ```
   wtdumprl -o DESC | more
   ```

 You should see the Net.Commander event classes in the display.

Table 16. Classes

Cause of event	Event Class
Unauthorized	NetscapeUnauth
Forbidden	NetscapeForbidden
Not Found	NetscapeNotFound
Server Error	NetscapeServerError

These are the derived classes from NetscapeWWW.

8. Integration of the events into the event console.

You may not have to make any changes here, if log file messages are already assigned to a group. The Netscape events are part of this larger group.

Tips and Tricks

If you have problems with the events, make sure that the event server is running properly. For this, please refer to Chapter 10, “Introduction to the TME 10 Enterprise Console” on page 319 and Chapter 11, “Tivoli/Enterprise Console Adapters” on page 355.
Check the user ID running the Netscape server and the user ID used for testing the shell scripts. Make sure that these are the same.
Check the permissions and the owner of the file /tmp/postemsg.out.
Run the shell scripts manually. You should get one event each time you execute one of the scripts. If it is not working, try testing the postemsg command itself.

14.4 Other Internet Servers

You can use the net.Commander to manage other Internet servers namely, mail servers or news (NNTP) servers. In fact, the support functions are not very different from the Web server management, so we do not describe them in detail here.

The following table shows the data you need to create mail and news Servers:

Mail Server

- Server Style: (only Sendmail supported)
- Config File (default: /etc/mail/sendmail.cf)
- Incoming Queue Directory (default: /var/spool/mail)
- Outgoing Queue Directory (default: /var/spool/mqueue)
- Log File (default: /var/adm/syslog/mail.log)
- Mail Master Mail Alias (default: postmaster)

News Server

- Server Style (default: Netscape, supported: Netscape and INN)
- News Directory (default: /usr/ns-home/news-563)
- Bin Directory (default: /usr/ns-home/bin/news)
- News Server Port Number (default: 563)
- News Master Mail Alias (default: newsmaster)
- Minimum Mb (desired) disk free for incoming news (default: 5)
- Minimum Mb (desired) disk free for outgoing news (default: 5)
- Minimum Mb (desired) disk free for active news (default: 5)
- Monitoring mode for this server: async./sync. (default: async.)
- Task mode for this server: async./sync. (default: async.)

14.5 Firewalls

Net.Commander also provides some support for firewall monitoring. Firewall configurations frequently consist of an IP filter and an application gateway. Often the IP filter is a dedicated IP router, which filters the IP packets. The application gateway machine, which is directly connected to the secure network, is usually a UNIX workstation, such as a Sun Sparc or IBM RS6000 workstation. Net.Commander provides a T/EC adapter so that everything logged by the firewall code or the UNIX syslog daemon can be sent as an event to the T/EC event server.

Net.Commander has a predefined configuration for the Sun Firewall-1 product, but currently does not support the IBM Internet Connection Secured Network Gateway.

There are two ways to configure the firewall machine to enable it to send events to the T/EC, as a TME managed node or not as a managed node. As a managed node you have to install the TME platform. This is additional, complex code on a very sensitive machine, which you may not want to allow. However, having the platform does mean that the events can be sent using a secure mechanism, with encrypted packet headers. If you install the log file adapter without first installing the machine as a TME managed node you avoid having the TME code running on your firewall system. However, the event path is a simple, unauthenticated TCP/IP socket connection. Also, you lose the ability to use other TME components, such as Sentry for monitoring firewall processes and utilization.

14.6 An Example of Extending the Log File Adapter to Monitor a Firewall

We have said (above) that Net.Commander does not currently provide support for the IBM Secure Network Gateway (SNG) firewall. This support will be developed at some time in the future. However, we decided to use it as a simple example of how the log file adapter can be configured to handle a different system environment.

In our example, the firewall gateway was an RS6000 with two token-ring adapters. We installed the log file adapter on this machine and configured it to capture all messages from SNG.

The scenario:

Firewall rs600014 (secure adapter address 9.24.104.191, unsecure adapter address 4.4.4.4)

Event Server rs600020 (in the secure network)

We did not install rs600014 as a TME managed node, so we followed the instructions for installing the log file adapter in an unsecured environment in the *Tivoli Enterprise Console Event Adapter Guide: Logfile*. The word “unsecured” in this case refers to the fact the it does not use TME secure remote procedure call for event delivery.

14.6.1 Preparation of the Firewall

The first thing is to set up the firewall properly, before the log file adapter can be installed:

1. Install the IBM Internet Connection Secured Network Gateway using the standard installp procedure.

 The version we used was V2R2 on the AIX 4.1.4 operating system.

2. Customize the firewall itself, including the syslog configuration. One option is to use the LAID package described in *Building a Firewall with the IBM Internet Connection Secured Network Gateway*, SG24-2577. Within this package there are installation scripts for a number of AIX subsystems that write further information to syslog, beyond the standard SNG support. For this example we installed the audit, authent, fw_log, and syslog subsystems.

14.6.2 Customizing the Log File Adapter for the Firewall

For a full discussion of what is involved in adding extra functions to the log file adapter, refer to 12.2, “Extending the Logfile Adapter to Manage a Distributed Application” on page 393.

In this case our task was easier because we were dealing with messages from syslog, not messages written directly by an application. This meant that our mapping logic from the message text into event classes was based on the standard mapping provided by the log file adapter. The configuration items we created were:

- New baroc definitions for event classes from our new message types
- Entries in the tecad_logfile.fmt file to map our additional message types into the new event classes

Figure 440 shows the baroc definition we created for the audit subsystem messages. The new class is called simply AUDIT and it contains slots for each of the elements present in an audit subsystem message.

```
TEC_CLASS :
        AUDIT ISA Logfile_Base
        DEFINES {
                audit_weekday:      STRING;
                audit_month:        STRING;
                audit_day:          STRING;
                audit_time:         STRING;
                audit_year:         STRING;
                audit_user:         STRING;
                audit_prog_type:    STRING;
                audit_type:         STRING;
                audit_retcode:      STRING;
                audit_file:         STRING;
                audit_out:          STRING;
        };
END

Events
```

Figure 440. Baroc Definition for Audit Subsystem

The steps to add the new class to the T/EC server are documented in Chapter 10, “Introduction to the TME 10 Enterprise Console” on page 319.

To use the new event class, you also have to assign message data to the baroc event slots. We updated the tecad_logfile.fmt file with the entries shown in

Figure 441 on page 453. This maps each element of the message into a different slot.

```
/*
 * AUDIT
 *
 */
FORMAT AUDIT FOLLOWS Logfile_Base
%t %s AUDIT: %s %s %s %s %s %s %s %s %s %s %s %s*
audit_weekday $3
audit_month $4
audit_day $5
audit_time $6
audit_year $7
audit_user $8
audit_prog_type $9
audit_type $10
audit_retcode $11
audit_file $12
audit_out $13
msg PRINTF("File %s is used (%s) from %s/%s. Exitstat: %s", audit_file,
 audit_type, audit_user, audit_prog_type, audit_retcode)
END
```

Figure 441. Logfile Format File Extension for the Audit Subsystem

To convert this .fmt file into a .cds file that the log file adapter can use, we used the gencds command as described in 12.2.3, “Mapping Message Formats to Event Classes” on page 396.

Chapter 15. Adding Function to the TME Desktop

TME provides a number of interfaces that can be used to extend the base function of the platform and of applications that use it. Some of these interfaces are what may be thought of as tight integration. That is, they are low-level programming APIs that provide access to the object request broker functions. Most APIs of this type are part of the Applications Development Environment (ADE) toolkit.

ADE is beyond the scope of this book, but there are a number of less demanding interfaces that can nevertheless give good integration and gain synergy by allowing an application to exploit the TME platform functions. We have already seen some examples of applications that use these interfaces in the /Plus modules and in Net.Commander. In this chapter we show three simple examples:

1. An example of using the Task Library Language (TLL) to create more complex and friendlier TME tasks and jobs (15.1, “Using the Task Library Language (TLL)”).
2. An example of modifying TME policy to limit the actions available in an application (15.2, “Configure a TME Policy Object for Task Library” on page 467).
3. An example of using the Application Extension Facility (AEF) to add functions to the TME 10 User Administration (Tivoli /Admin) application (15.3, “Using the Tivoli Application Extension Facility (AEF)” on page 479)

15.1 Using the Task Library Language (TLL)

TLL is the language used to define task libraries, tasks and jobs to TME. We have described how to create these in Chapter 4, “Task Libraries, Tasks and Jobs” on page 81. The examples we used in that case were quite simple and just used the standard task dialogs and panels. However, the standard process does not allow you to define a task that takes input in the form of arguments. In this section we show how to use TLL to create such a task.

If you plan to create more complex tasks of this sort, you should refer to the *Tivoli/Task Library Language Developer's Guide*.

15.1.1 Getting Started

We found that a simple way of becoming familiar with the format of a task library description file was to export the Show_Filesystems task we had already created (see 4.2.3, “Create a Task” on page 91). Although this did not show us any examples of setting up argument layouts, it gave us a starting point where we could be sure that the syntax was correct.

To export an existing task library:

```
wtll -F /tmp/General_Tasks.tar -l General_Tasks
```

This command unloads the library into tar file /tmp/General_Tasks.tar. Note that although the policy region role needed to create task libraries is senior, this did not allow us to use the wtll command to export the task library. For that we needed to set a TMR Role of admin.

© Copyright IBM Corp. 1996

The tar file has two files packed into it, as shown in Figure 442 on page 456.

```
tar -tvf /tmp/General_Tasks.tar

drwxr-xr-x 464 114       0 Aug 13 10:24:46 1996 .
-rw-r--r-- 464 114   12278 Aug 13 10:24:46 1996 ./0.default
-rw-r--r-- 464 114    2327 Aug 13 10:24:46 1996 ./tll
```

Figure 442. Listing the Contents of the Exported Task Library

We unloaded this tar file into our own directory, using the following commmands:

```
cd /u/lynn/tll_General_Tasks
tar -xvf /tmp/General_Tasks.tar
```

Now we can look at the two files in a little more detail. The file 0.default is the code that is executed by the task, so in this case it is actually a copy of the df command (/usr/bin/df). The file tll contains the definition of the task library and its tasks. Figure 443 shows the tll file from our General_Tasks example.

```
#ifndef TASK_BINDIR
#define TASK_BINDIR "./"
#endif
TaskLibrary "General_Tasks" {
        Context = ("_!_","*",1);
        Distribute = ("_!_","ALI",1);
        HelpMessage = ("_!_","Conventional Task Library",1);
        Requires = ("_!_",">2.5",1);
        Version = ("_!_","1.0",1);


        Task Show_Filesystems {
                Description = ("_!_","Upgraded Task",1);
                HelpMessage = ("_!_","No Help Available",1);
                Uid = ("_!_","*",1);
                Comments = ("_!_","Task Name  :
General_Tasks/Show_Filesystems
Task Created         : Fri Aug  9 11:27:56 1996
Task Created By      : lynn@rs6000024.itso.ral.ibm.com
Task Files
    default          rs6000024         /usr/bin/df
Distribution Mode    : ALI
Task Comments        :
Runs the \"df\" command to Show Filesystems
-----------------------------------------
Task Modified        : Mon Aug 12 15:27:28 1996
Task Modified By     : lynn@rs6000024.itso.ral.ibm.com
Task Files
    default          rs6000024         /usr/bin/df
Distribution Mode    : ALI
Task Comments        :

-----------------------------------------
",1);
                Roles = ("_!_","user",1);
                Implementation ("default") Binary TASK_BINDIR
                 "0.default";
        };

}
```

Figure 443. Exported Task Library Definition

Having exported this task library, one of the things you can do with it is to import it into another TMR. We decided to try doing this. If you have connected TMRs there is no need to copy task libraries in this way, but it is a useful method of propagating your task libraries where they are not connected and it is also an

important technique if you are creating a package of TME function for other people to use (as in the case of a /Plus module, for example).

To import the task library into policy region Prod in a separate TMR, we performed the following actions:

1. Copy the tll and 0.default files to the new TME server.
2. From an authorized administrator's command line, enter:

   ```
   wtll -p Prod -P /usr/ccs/lib/cpp /u/lynn/General_Tasks/tll
   ```

 In this case we expected that the administrator issuing the wtll command would need the same authorization as it would to create a new task library, that is, the policy region senior role. In fact we discovered that a TMR role of admin was required to run the command.

 When you import a task library you may see this message:

   ```
   Tue Aug 13 14:00:29 EDT 1996 (2): operation `unable to retrieve task
   file rs600011:/home/lynn/./0.default' failed
   ```

 We found that we had to run the wtll command from the directory where the tll and 0.default files existed, in this case directory /u/lynn/General_Tasks.

15.1.2 Create a Task Library Using TLL

The next objective was to use the exported TLL as a model to create our own task library description that allows you to specify arguments to be passed to the task.

Our objective is to create a task that modifies the NetView for AIX object database, as follows:

- The name of the task library is NetView_Tasks.
- We create a task in this library called Set_isTME_NODE. This task runs a command to update the NetView database, setting a capability field isTME_NODE to True or False for selected hosts.
- The task requires three arguments:
 1. The names of the nodes for which the NetView field is to be set
 2. The name of the NetView capability field
 3. A value of True or False

To implement this plan we need to create a new task library description file. Figure 444 on page 458 shows the tll file that we produced.

```
TaskLibrary "NetView_Tasks" {
        Version = "1.0";
        Distribute = ("_!_","ALI",1);
        Requires = ("_!_",">2.5",1);
        HelpMessage = ("_!_","Conventional Task Library",1);
        Context = ("_!_","*",1);

        ArgLayout Host {
             TextChoice Program {
                Implementation ( "default" )
                  .#!/bin/ksh
                  .hosts=`awk < /etc/hosts '/*[0-9][0-9]*/ {print $2}' `
                  .for i in $hosts
                  .  do
                  .     choices="$choices `host $i   cut -d \" \" -f1`"
                  .  done
                  .for i in $choices
                  . do
                  .  echo $i
                  . done
                  ;

             };
        };

        ArgLayout Capability {
             ChoiceButton Program {
                Implementation ( "default" )
                  .#!/bin/ksh
                  .echo "isTME_NODE"
                  ;

             };

        };

        ArgLayout TrueFalse {
             RadioButton { {("_!_","True",1) "True"}
                {("_!_","False",1) "False"} };

        };

Task Set_isTME_NODE {
        Description = (_xxx_,"Set isTME_NODE",1);
        HelpMessage = (_xxx_,"No help available",1);

        Argument (_xxx_,"Hostname",1) {
             Layout = "Host";
        };


        Argument (_xxx_,"Capability",1) {
             Layout = "Capability";
        };


        Argument ("_!_","True/False",1) {
             Layout = "TrueFalse";
        };

       Roles = ("_xxx_","user",1);
       Implementation ( "default" ) Binary "/usr/local/bin/set_tme_field";
        };

}
```

Figure 444. Task Library Definition for NetView_Tasks

This definition uses three GUI input devices, a text list, a text choice button and a set of radio buttons to solicit the data needed by the program. Descriptions of the task library header, argument layouts and task definitions can be found in the *Task Library Language Developer's Guide*, so we do not describe the

contents of the file in detail here. However, you may find the following comments useful in addition to the guidance in the guide:

- In our example, we are not using the Message Catalog facility, but we are using several different forms of the simulated message catalog described in the *TLL Developer's Guide*:

  ```
  RadioButton { {("_!_","True",1) "True"}

  Description = (_xxx_,"Set isTME_NODE",1);

  Roles = ("_xxx_","user",1);
  ```

 It seems that "_!_", _xxx_ and "_xxx_" are all valid, although labels containing special characters need to be enclosed in quotes. There seems to be no problem in mixing the format or having different dummy names within the same file.

- The value following Implementation is a list of the supported platforms for the shell script or program being run. This is equivalent to the Platforms Supported list when setting up a task on the desktop. Note that the value *default*, equates to generic.

- Shell scripts built in to the file must start each line with a . (period). In the example we use this capability in the TextChoice field for ArgLayout Host. This generates a scrollable list of selectable text fields, in our case, hostnames. The script example here gets the second field from /etc/hosts, runs the host command and cuts at the first space, to give us the full DNS name if that is applicable.

- ArgLayout Capability is only supplying one item, isTME_NODE. We have included this field, even though there is really not a choice in what the value can be, as the program we are eventually executing requires this argument to be passed as the second parameter. We display the field here, so that it will be passed correctly.

- Roles defines the roles required to execute the task, in this case user.

- Implementation within the task definition is calling an external program called /usr/local/bin/set_tme_field. This program must exist on the system where the wtll command is being run. There does not appear to be a facility to specify a hostname where the executable can be found.

 If you have different executables for multiple platforms, you can specify multiple Implementation lines, for example:

  ```
  Implementation ( "aix3-r2" ) Binary "/usr/local/bin/aix32code";
  Implementation ( "solaris2" ) Binary "/usr/local/bin/solaris2code";
  ```

15.1.2.1 Compiling the Task Library Definition

Before it can be used, the task library description must be compiled. The command to perform this is as follows:

```
wtll -p Prod -P /usr/ccs/lib/cpp NetView_Tasks
```

This will create the task library in policy region Prod, using the C preprocessor /usr/ccs/lib/cpp. You could also append the C preprocessor to $PATH. If you want to recreate an existing task library, specify the -r flag, for example:

```
wtll -r  -p Prod -P /usr/ccs/lib/cpp NetView_Tasks
```

Unless you are an extremely lucky person, you will probably not get the TLL source code exactly right the first time you try to compile it, in which case you will be greeted by some syntax error messages such as:

```
cpp: 1501-228 input file NetView_Tasks not found
Syntax errors:
tll error in Sample_Tasks2, line 1: Improper task library statement
(missing "TaskLibrary" keyword; saw <ugliness> instead)
```

or alternatively:

```
Syntax errors:
tll error in Sample_Tasks2, line 1: Cannot run preprocessor "cpp"
Improper task library statement (missing "TaskLibrary" keyword;
saw Cannot instead)
```

In fact, both of these messages were caused by the same error, a missing ; (semi-colon) at the end of the TaskLibrary statement. The different syntax error messages were due to different command invocations. In the first case, -P /usr/ccs/lib/cpp was included in the command; in the second case /usr/ccs/lib/cpp was included in the PATH.

Whichever version of the message you prefer, it does not give a lot of help in pointing at where the problem lies! The approach we found best is to strip the TLL definition down to a minimum and then gradually rebuild it until you find the error, so a possible sequence is:

1. Copy your file to save it.
2. Remove task definitions from the file and try compiling with just the task library definition.
3. Get the file to a stage where it works, then gradually add back the task definitions.

When the file has compiled successfully you will see the task library and the task added within the policy region, as shown in Figure 445 and Figure 446 on page 461.

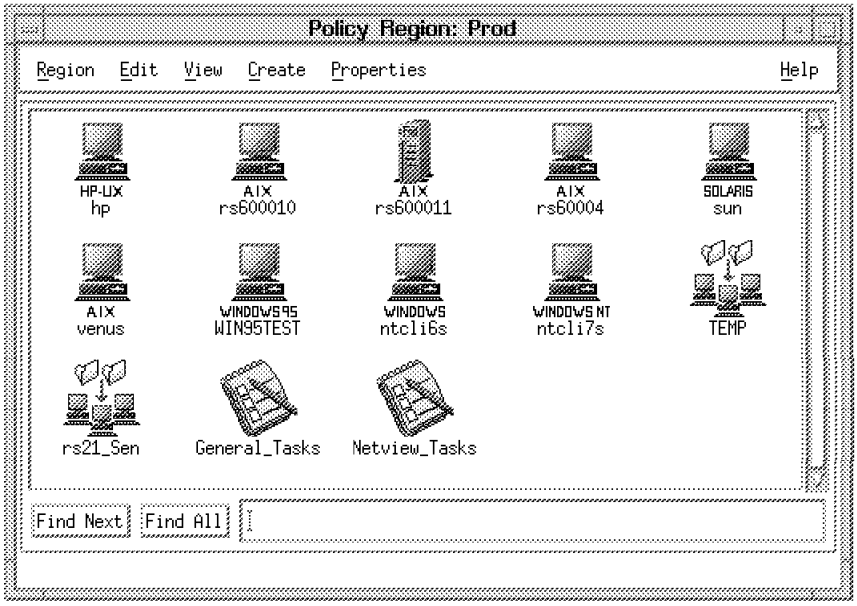


Figure 445. The New Task Library Appears

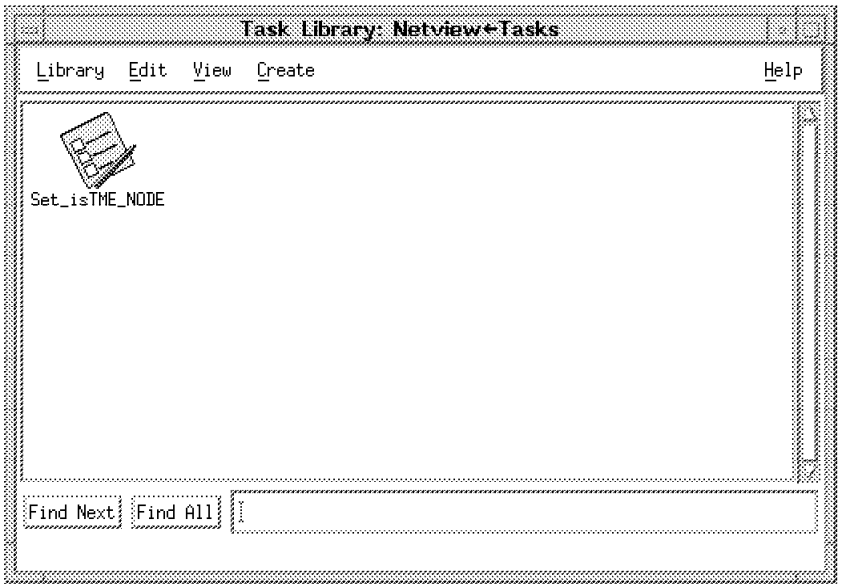


Figure 446. The New Task Library Has a Task in It

15.1.3 Executing a Task with Arguments from the Desktop

Now you can try to execute the task, by double-clicking the icon. You will see the standard Execute Task window, as shown in Figure 447 on page 462, in which you specify the execution target machine and the destination for output.

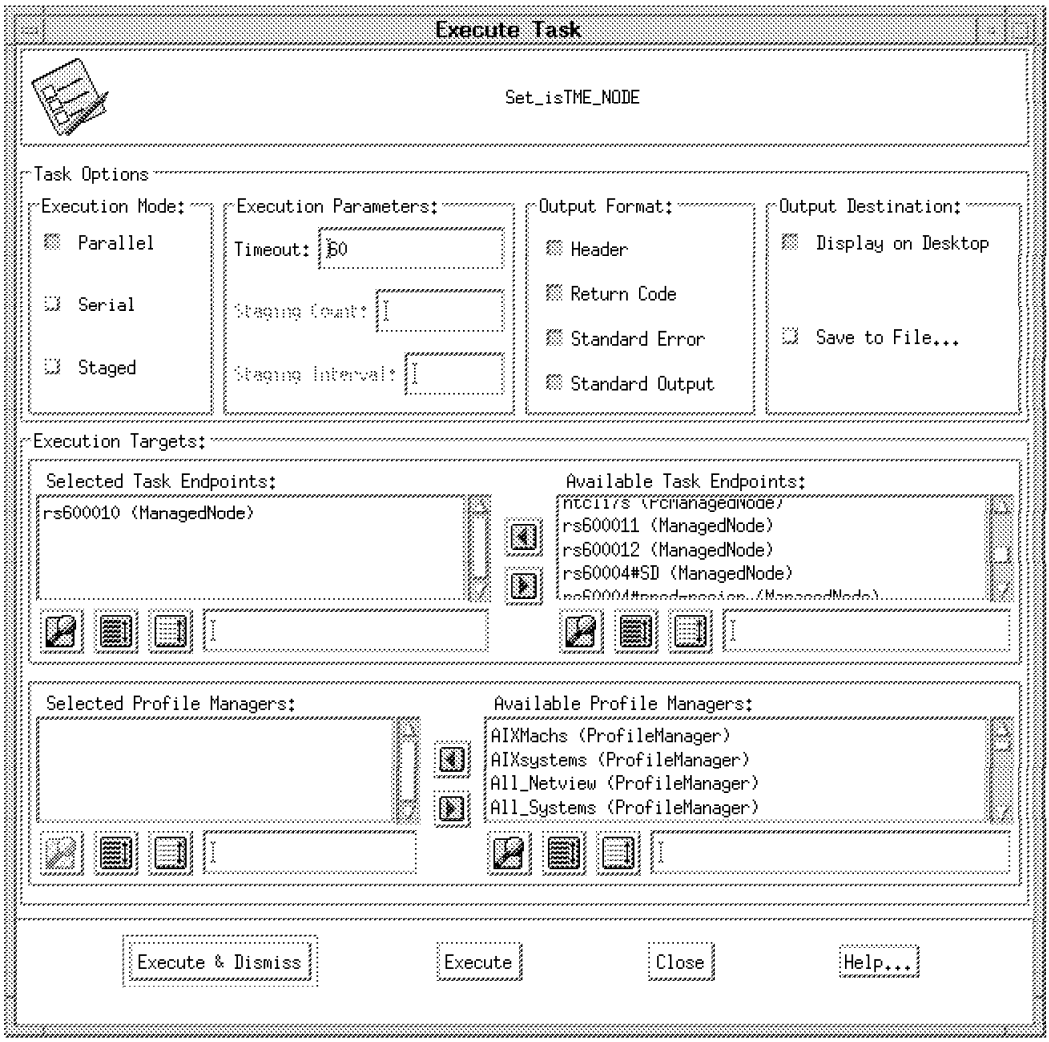


Figure 447. Task Execution Dialog

In this particular example, the command we are running must execute on a machine running NetView, so we chose rs600010 as the execution target. When you click on **Execute**, you will be presented with the window created by the TLL code, in which to enter the arguments for the command (see Figure 448 on page 463).

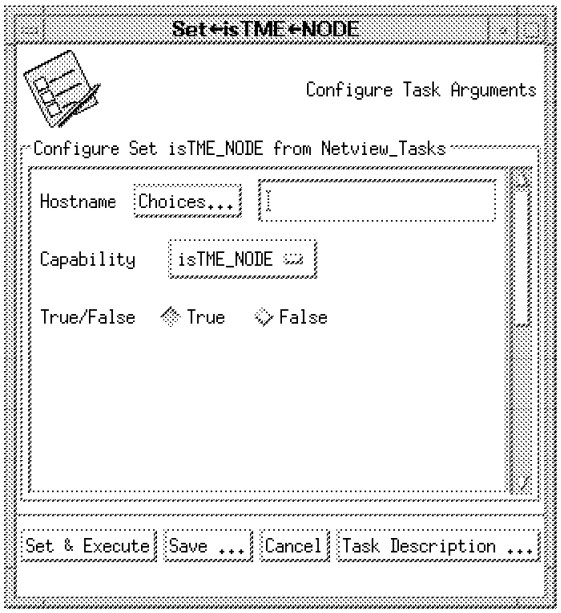


Figure 448. Dialog Prompts for Command Input

The Hostname can be entered manually, or selected from a list of choices. Note that the hostname we are choosing here is not a necessarily a TME managed node; it is just the name of a host in the NetView database.

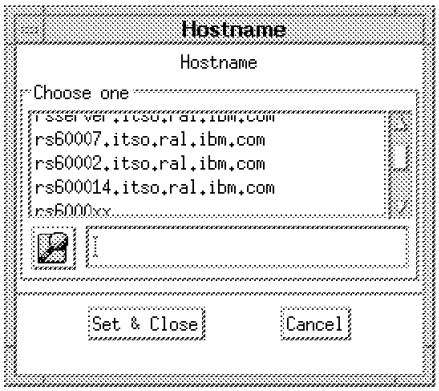


Figure 449. Choice List for Hostname

1. The choice list was produced from the shell script built in to the task to query /etc/hosts.
2. The capability has only the one choice of isTME_NODE as this was all that was specified in the source code.
3. True/False appears as a radio button.

This task can also be run from the command line:

```
wruntask -h rs600010 -l NetView_Tasks -t Set_isTME_NODE \
         -a rs600011.itso.ral.ibm.com -a isTME_NODE -a True
```

You will not see the pop-up window to enter your arguments when running from the command line.

15.1.4 Create a Job to Execute Set_isTME_NODE

Whenever you execute a task, you have to select an execution target where the task is to run. If tasks are always going to execute on the same targets, you can fix the target names by creating a job. The target can be any combination of valid TME subscribers, that is, managed nodes and/or profile managers.

In this case, task Set_isTME_NODE must execute on a machine with NetView for AIX installed. In our environment we had two machines running NetView, rs600010 and rs600021. However, initially only rs600010 was available; we planned to add rs600021 later. If we set up the job to execute on a specific task endpoint of rs600010 we will need to edit the job to add rs600021 later. Instead, we will show how to set up the job with an execution target of a profile manager where the profile manager has subscribers of all NetView machines. This will mean that the job will execute on any new NetView machine that subscribes to the Profile Manager, without having to edit the job.

Create the Profile Manager

We will create a profile manager called AIX_NetView:

1. Open the Prod policy region from the desktop.
2. Check that Prod has ProfileManager in its current managed resources.

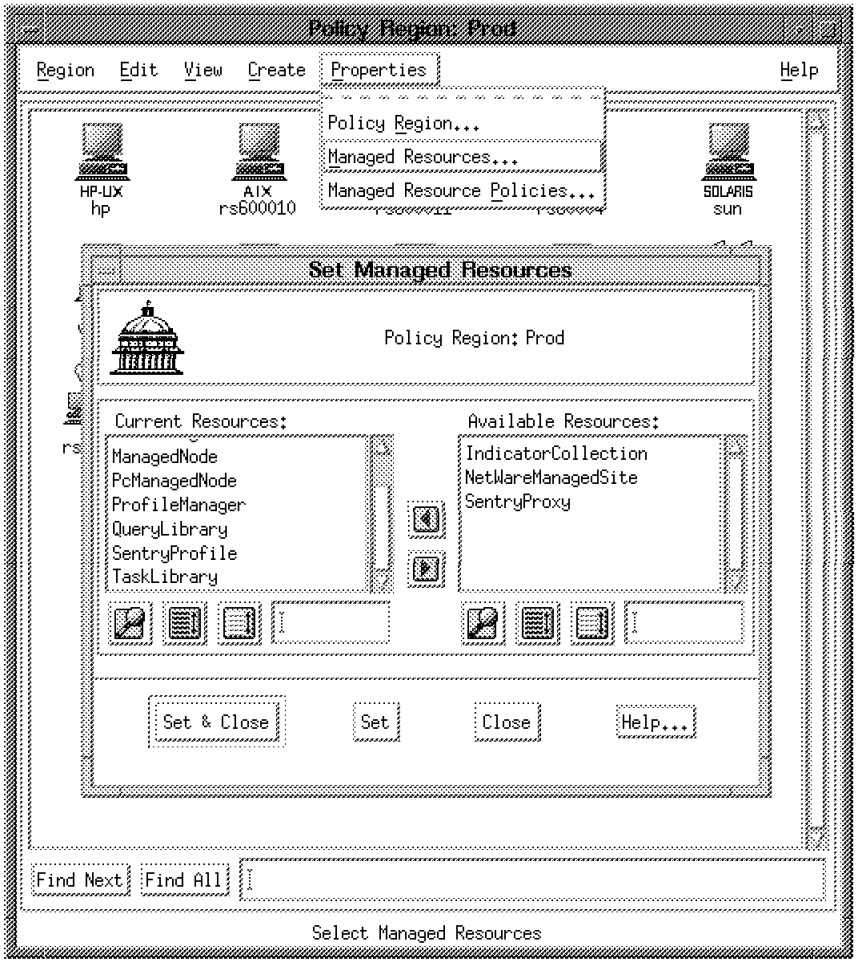


Figure 450. Check Managed Resources

3. Create a new profile manager, byy selecting **Create** and then **Profile Manager** from the menu bar.

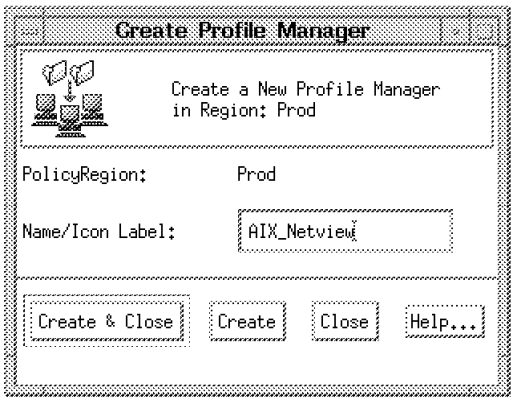


Figure 451. Create a Profile Manager

4. Add subscribers to the profile manager, by clicking on the profile manager icon with the right mouse button and selecting targets from the list of nodes.

Create a Job

We now create a job from the task Set_isTME_NODE that we created earlier. A job is simply a task for which the execution target(s) and options have already been defined. In this case we define the profile manager that we have just created as the target for the job. This means that if we want to add further execution targets, we just have to subscribe them to the profile manager, instead of updating all of the jobs and other profiles that are affected by the addition of the node.

A task must exist before a job can be created from it.

1. Open task library NetView_Tasks from the Prod policy region.
2. Select **Create** and then **Job...** from the menu bar. Enter the name of the job. It can be the same as the task name.
3. Select the execution target. In our case it is the profile manager AIX_NetView as shown in Figure 452 on page 466.

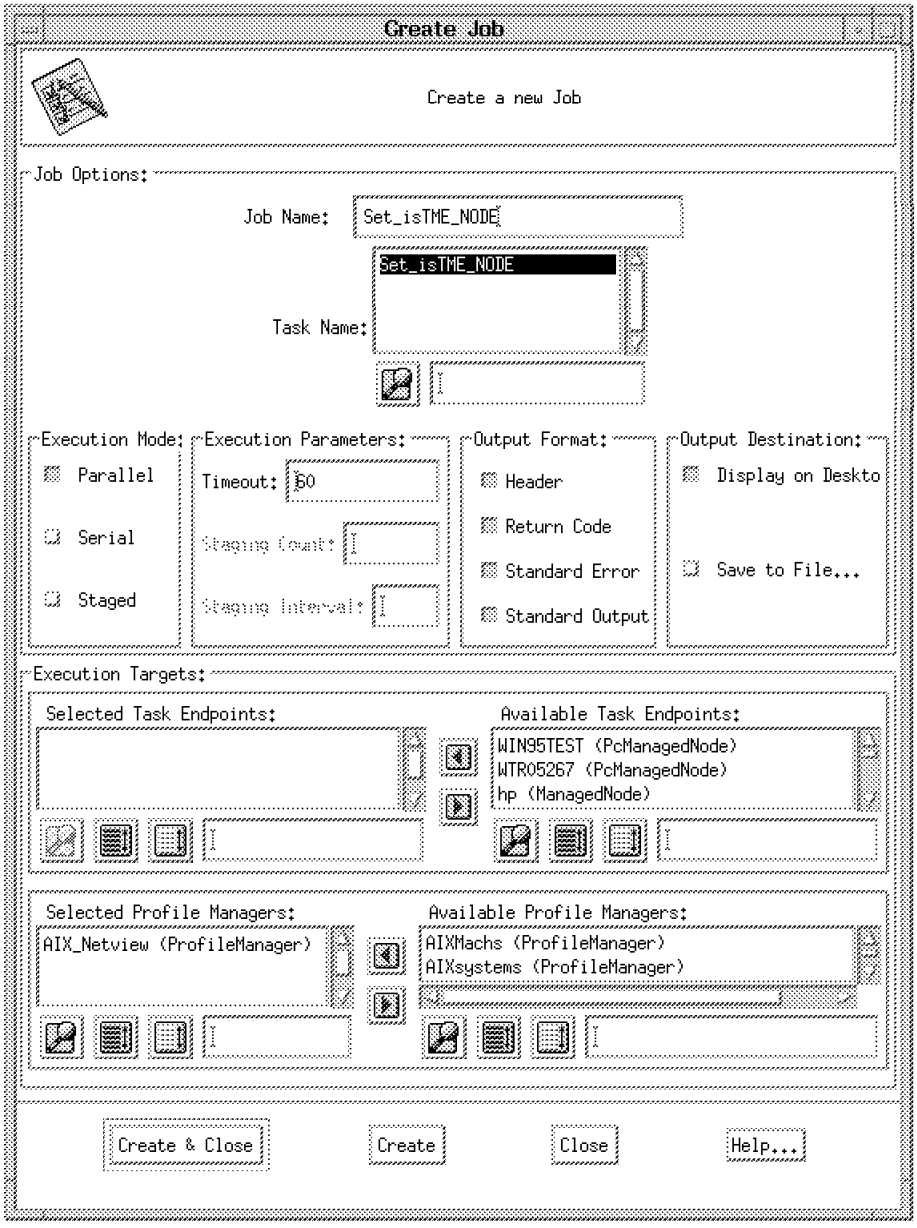


Figure 452. Define Execution Targets for Job

4. Select **Create & Close**.

 The icon for Job Set_isTME_NODE will appear in the task library window.

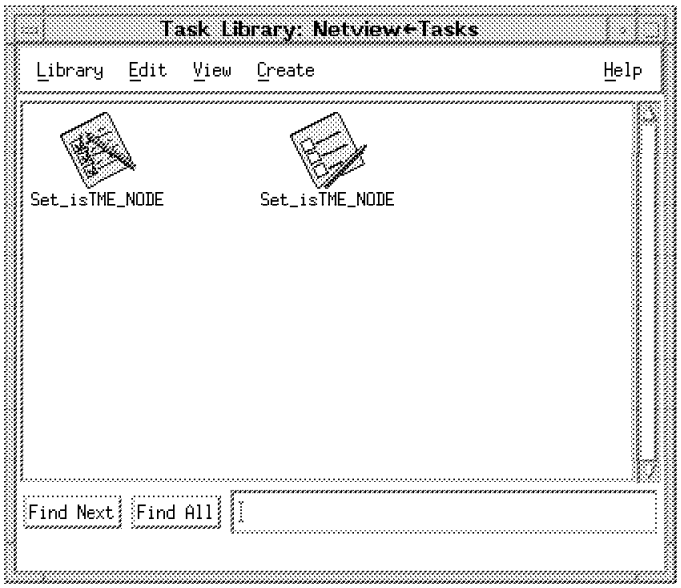


Figure 453. Job and Task Icons in a Task Library

15.1.4.1 Create the Profile Manager and Job from the Command Line

1. Create a profile manager:

   ```
   wcrtprfmgr Prod AIX_NetView
   ```

2. Subscribe TME resources, for example, managed nodes, to a profile manager:

   ```
   wsub @AIX_NetView @ManagedNode:rs600010
   ```

 or

   ```
   wsub @AIX_NetView @ManagedNode: rs600010 @ManagedNode:rs600021
   ```

3. Create a job in a task library:

   ```
   wcrtjob -j Set_isTME_NODE -l NetView_Tasks -t Set_isTME_NODE \
           -M parallel -m 60 -o 17 -p AIX_NetView -D
   ```

15.2 Configure a TME Policy Object for Task Library

This example builds on the task library example in the previous section. We have already mentioned that the execution target for the task Set_isTME_NODE must be a machine that is running NetView. The way in which we have created the task under policy region prod means that this task library will be subject to whatever default policies are in effect in that region. This means that there is nothing to prevent us from subscribing a machine that is *not* running NetView to the profile manager. Often this is not a problem, because you can rely on the experience of the user not to make an invalid subscription. However, it is quite simple to invoke a new policy object that limits the list of available subscribers.

We now create a new policy specifically for the NetView_Tasks that will tailor the lists of available endpoints and available profile managers, so that only NetView machines can be selected. See the *Tivoli Management Platform User's Guide* to gain a general understanding of a policy.

We now complete the following steps:

1. Look at the existing policies in Prod.
2. Create a new policy object for TaskLibrary.
3. Change policy methods for the new policy object.
4. Move the task library NetView_Tasks into a separate region.
5. Apply the new policy to NetView_Tasks.

15.2.1.1 Look at the Existing Policies in Prod

Each resource managed by a policy region has a default policy associated with it. The task library default policy is called BasicTaskLibrary.

1. Select option **Managed Resource Policies** from the Properties pull-down menu in the Prod policy region.

 Look at the default policy for managed resource TaskLibrary.

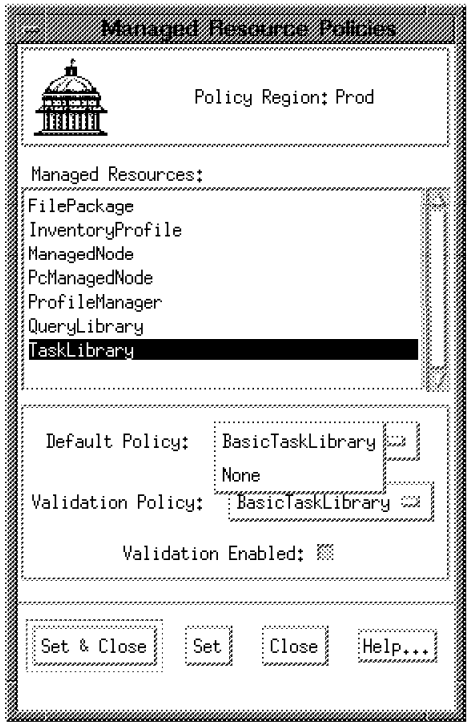


Figure 454. Selecting Managed Resource Policies

The list should include BasicTaskLibrary and None. If you set the Default Policy to None, you will no longer be able to create objects of that type in the policy region.

15.2.1.2 Create a New Policy Object for TaskLibrary

New policy objects can be created only from the command line with the wcrtpol command. They cannot be created from the Desktop. Enter the command to create a policy object called NetViewTasks:

```
wcrtpol -d TaskLibrary NetViewTasks
```

To list policy objects for resource TaskLibrary, enter:

```
wlspol TaskLibrary
```

You will see:

```
BasicTaskLibrary
NetViewTasks
```

15.2.1.3 Change Policy Methods for the New Policy Object

To list the policy methods for the resource TaskLibrary, enter:

```
wlspolm TaskLibrary
```

You will see:

```
tl_def_dist_mode
tl_def_man_nodes
tl_def_prof_mgrs
tl_def_set_gid
tl_def_set_uid
```

These are described in the *Tivoli Management Platform User's Guide*, under Task Library Policy.

We want to change the rules governing the lists of available endpoints and available profile managers, so we will need to change:

```
tl_def_man_nodes
tl_def_prof_mgrs
```

The methods associated with BasicTaskLibrary are copied to the new Policy object NetViewTasks as it is created.

To retrieve the body of the policy methods for NetViewTasks, enter:

```
wgetpolm -d TaskLibrary NetViewTasks tl_def_man_nodes > /u/lynn/man_nodes
wgetpolm -d TaskLibrary NetViewTasks tl_def_prof_mgrs > /u/lynn/prof_mgrs
```

The tl_def_man_nodes method is a script shown in Figure 455.

```
#!/bin/sh
library=`wlookup Library`
supporters=`idlcall $library select_instance_managers \
'\"TMF_CCMS::ProfileEndpoint\"' FALSE FALSE FALSE`

num_sup=`idlarg 1 $supporters`
limit=`expr $num_sup + 1`
i=2
while [ $i -le $limit ]; do
        arg=`idlarg $i $supporters`
        name=`idlarg 2 $arg`
        name=`echo $name   tr -d \"`
        wlookup -r $name -a 2> /dev/null
                awk -F' ' '{
                        for (i = 1; i < NF; ++i) {
                                printf "%s (%s)", $i, "'"$name"'";
                                if (i < NF-1)
                                        printf "\t";
                        }
                        printf "\n";
                        }'
                i=`expr $i + 1`
done
```

Figure 455. The tl_def_man_nodes Method Script

The tl_def_prof_mgrs method is also a script (see Figure 456 on page 470).

```
#!/bin/sh
library=`wlookup Library`
supporters=`idlcall $library select_instance_managers \
'\"TMF_CCMS::ProfileManager\"' FALSE FALSE FALSE`

num_sup=`idlarg 1 $supporters`
limit=`expr $num_sup + 1`
i=2
while [ $i -le $limit ]; do
        arg=`idlarg $i $supporters`
        name=`idlarg 2 $arg`
        name=`echo $name   tr -d \"`
        wlookup -r $name -a 2> /dev/null
                awk -F' ' '{
                        for (i = 1; i < NF; ++i) {
                                printf "%s (%s)", $i, "'"$name"'";
                                if (i < NF-1)
                                printf "\t";
                        }
                        printf "\n";
                }'
        i=`expr $i + 1`
done
```

Figure 456. The tl_def_prof_mgrs Method Script

These two scripts need to be modified or completely replaced to change the rules as follows:

- tl_def_man_nodes will return only machines running NetView.
- tl_def_prof_mgrs will return profile managers that have only NetView subscribers.

In order to find out which machines actually have NetView installed, we did the following:

- Created a task in General_Tasks called Check_if_NetView_installed.

 This task executes a script called /usr/local/scripts/isNetView.sh:

```
#!/bin/ksh

ls /usr/OV > /dev/null

if [[ $? -eq 0 ]]; then
  echo "NetView is installed on this machine"
else
  echo "NetView is not installed on this machine"
  exit 1
fi
exit
```

 When this task executes, it checks for the existence of the NetView directory /usr/OV and will return an appropriate return code.

- We need to execute this task on all the managed nodes that would normally be available endpoints and filter out all those that are not NetView machines.

 This is the modified tl_def_man_nodes script:

```
#!/bin/sh
nodes=`wlookup -r ManagedNode -a   awk '{print $1}'`
  for i in $nodes
    do
      wruntask -l General_Tasks -t Check_if_NetView_installed -h $i \
        -o 2   grep Return   grep -w 0 > /dev/null
      if [[ $? -eq 0 ]]; then
        netviewnodes="$netviewnodes $i"
      fi
    done

for i in $netviewnodes
  do
    echo $i
  done
```

- We inserted code to run the task Check_if_NetView_installed on all managed node names that were found.

 The flag -0 2 on the wruntask command specifies that only the return code is required. The return code is checked for a value of 0 which will indicate that NetView is installed on that machine.

 This is the modified tl_def_prof_mgrs script:

```
#!/bin/sh

profs=`wlookup -r ProfileManager -a    awk '{print $1}' \
         grep NetView`

for i in $profs
  do
    echo $i
  done
```

 This code gets all profile manager names and searches for NetView as part of its name. This assumes that you have set up your profile managers following a naming standard that include NetView in the name of all those that have NetView subscribers.

These scripts can be tested manually from the command line to make sure they return the values you expect.

To replace the default method with these new scripts, enter:

```
wputpolm -d TaskLibrary NetViewTasks tl_def_man_nodes < /u/lynn/def_nodes

wputpolm -d TaskLibrary NetViewTasks tl_def_prof_mgrs < /u/lynn/prof_mgrs
```

At this stage the new policy object is not associated with any policy region, but if we change the default policy of Prod to use NetViewTasks the rules will apply to all tasks that execute, not just the NetView tasks.

We have to make the following changes to implement this policy just for NetView_Tasks:

1. Create a Subregion in policy region Prod, called NetView.

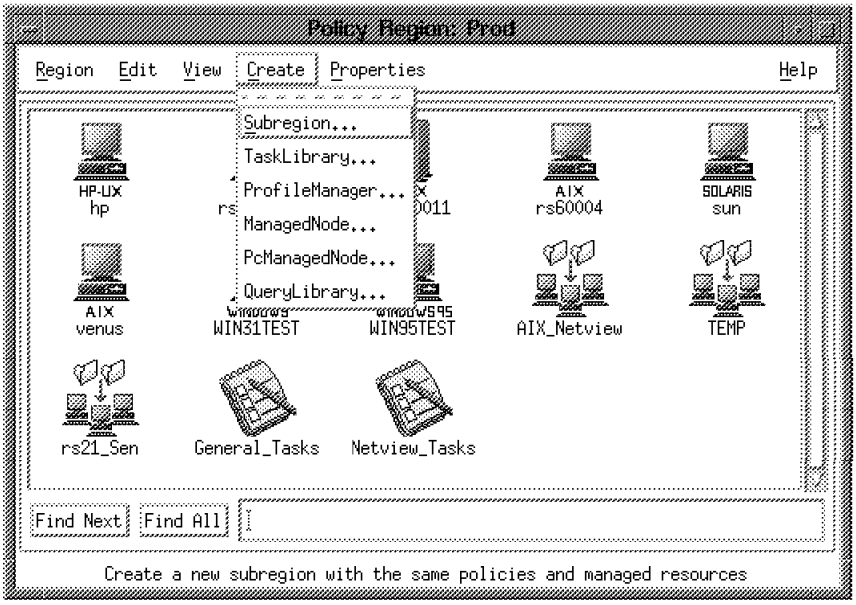


Figure 457. Create a New Policy Region

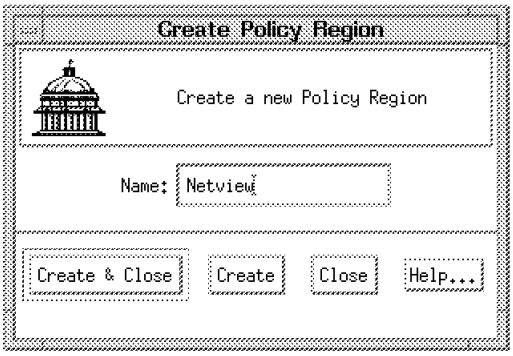


Figure 458. Create a New Policy Region

The new policy region will appear on Prod's window.

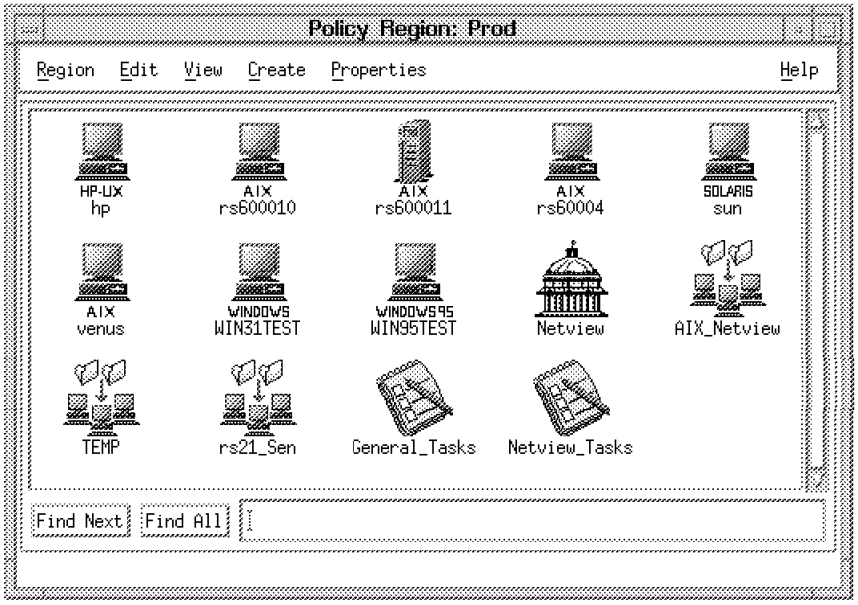


Figure 459. New Policy Region Icon

2. Double click the icon for policy region **NetView**.
3. Select **NetView_Tasks** from Prod. Move it to the NetView policy region by holding the Shift key, dragging and dropping the icon with the left mouse button onto the NetView policy region window.
4. Select **Managed Resource Polices** from the Properties pull-down menu on NetView.

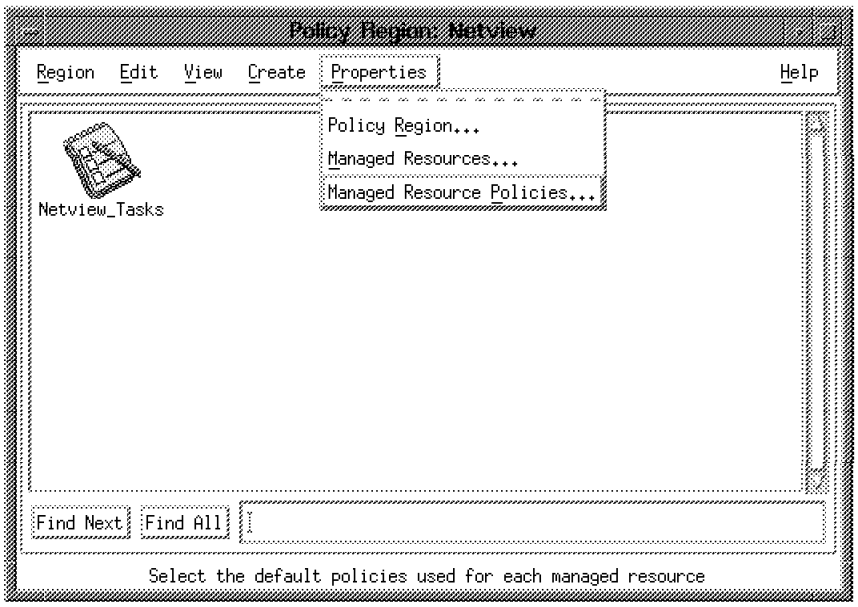


Figure 460. Task Library Moved to New Policy Region

5. Set the Default Policy for TaskLibrary to NetViewTasks.

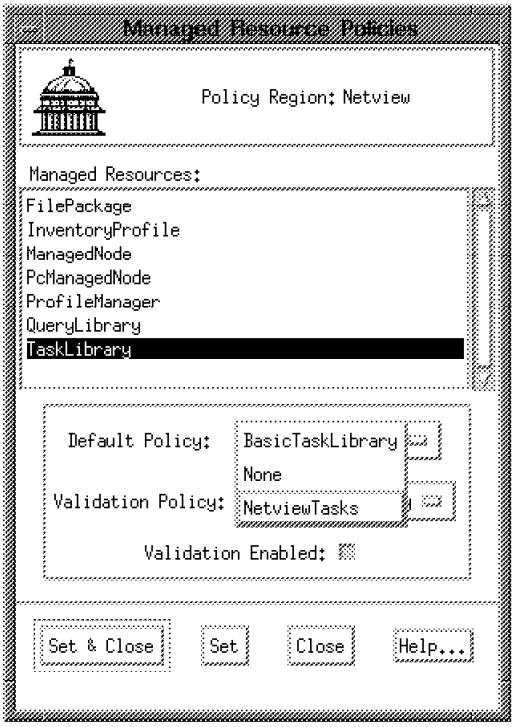


Figure 461. Invoke the New Policy Object

6. Select **Set & Close**.

Now execute the task Set_isTME_NODE in NetView_Tasks. The available task endpoints and available profile managers should now list only those with NetView.

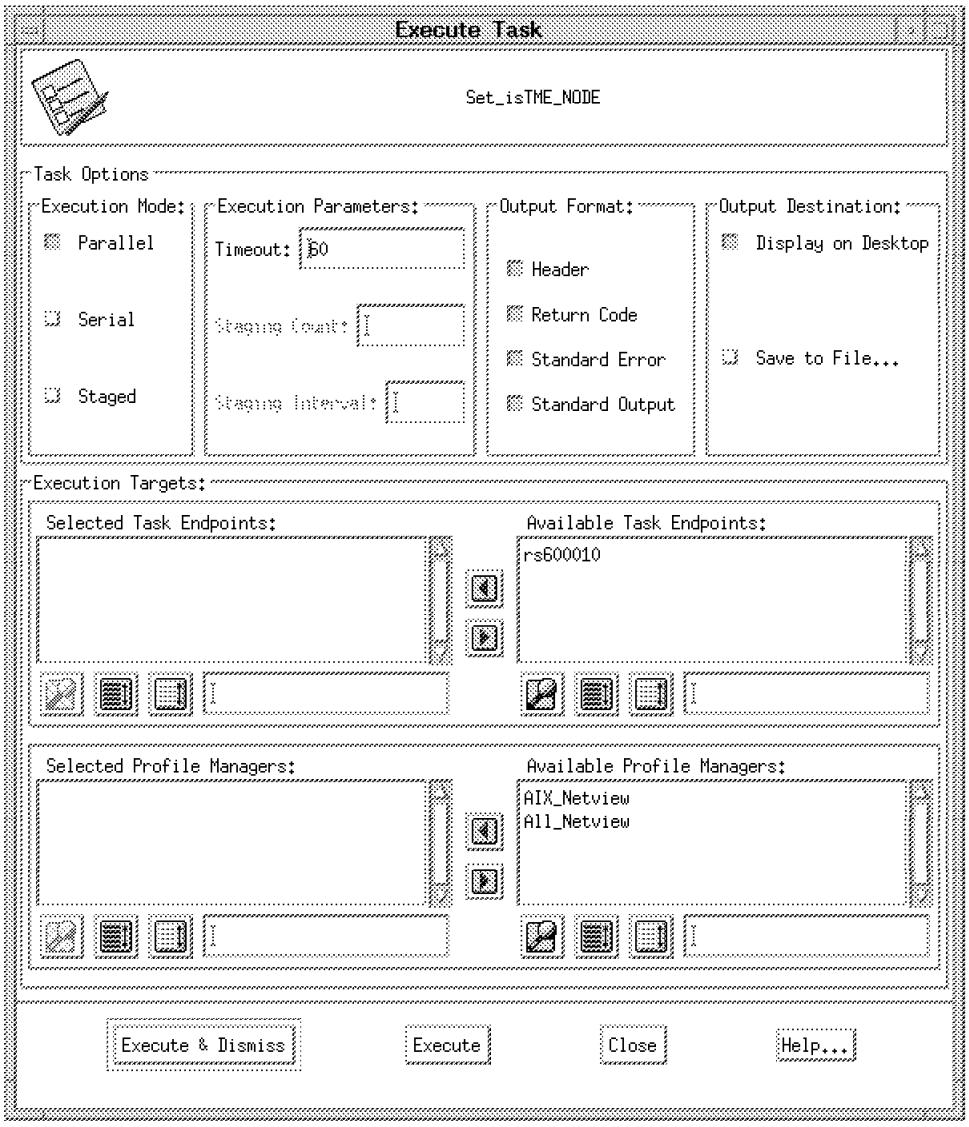


Figure 462. Reduced Target List Generated by New Policy Object

The tasks in the General_Tasks library in the Prod region will still use the BasicTaskLibrary methods.

15.2.2 Validation Policy

The new default policy NetViewTasks associated with the policy region NetView, will provide a tailored list of available endpoints and available profile managers when tasks are run from the desktop.

However, if tasks are executed with the wruntask command from the command line, this policy will not apply.

To ensure that the rules are followed, it is necessary to create a validation policy to check where the tasks will execute. The rules in the validation policy will generally be the same as those defined in the default policy.

The process for setting up a validation policy is basically the same as for a default policy:

1. List validation policies currently set up:

   ```
   wlspol -v TaskLibrary
   ```

 You will see:

   ```
   BasicTaskLibrary
   ```

2. Create a validation policy:

   ```
   wcrtpol -v TaskLibrary NetViewTasks
   ```

3. List the methods associated with the validation policy:

   ```
   wlspolm -v TaskLibrary
   ```

 You will see:

   ```
   tl_val_man_nodes
   tl_val_prof_mgrs
   tl_val_set_gid
   tl_val_set_uid
   ```

4. Retrieve the methods for validating task endpoints and profile managers:

   ```
   wgetpolm -v TaskLibrary NetViewTasks tl_val_man_nodes > /u/lynn/val_nodes

   wgetpolm -v TaskLibrary NetViewTasks tl_val_prof_mgrs > /u/lynn/val_mrgs
   ```

 These files will look like the following:

```
#!/bin/sh
##############################################################################
#
# $Id: tl_val_man_nodes.sh,v 1.2 1994/09/09 15:41:23 paul Exp $
#
# This script implements the "validate_execution_managed_nodes" policy
# method for the Task Library.  The script is provided with the name of
# the task, the label of the Admin and all of the managed nodes selected
# for execution targets of the task.  Modify the code below
# if you want something different returned.
#
# To debug your chanegs you could add the lines:
#
#            set -xv
#            exec > /tmp/debug.output 2>&1
#
# These lines will allow you to see any errors that occur by looking
# in the /tmp/debug.output file.
#
#  NOTE:  This script can also be called when a check_policy
#         operation is performed.  In that case, the name of the
#         Admin will be "any".  Make sure that you handle that case
#         if you modify this script.
#
##################################################################

task_name=$1
administrator=$2
shift 2

## Example of how to validate the list of managed nodes. ##

# for i in $*; do
#       if [ $i = "the evil managed node" ]; then
#          echo FALSE
#          exit 0
#       fi
# done

echo TRUE
exit 0
```

```
#!/bin/sh
#####################################################################
#
# $Id: tl_val_prof_mgrs.sh,v 1.2 1994/09/09 15:41:25 paul Exp $
#
# This script implements the "validate_execution_profile_managers" policy
# method for the Task Library.  The script is provided with the name of
# the task, the label of the Admin and all of the profile managers
# selected for execution targets of the task.  Modify the code below
# execution targets of the task.  Modify the code below
# if you want something different returned.
#
# To debug your changes you could add the lines:
#
#               set -xv
#               exec > /tmp/debug.output 2>&1
#
# These lines will allow you to see any errors that occur by looking
# in the /tmp/debug.output file.
#
#  NOTE:        This script can also be called when a check_policy
#               operation is performed.  In that case, the name of the
#               Admin will be "any".  Make sure that you handle that case
#               if you modify this script.
#
#
#####################################################################

task_name=$1
administrator=$2
shift 2

## Example of how to validate the list of profile managers. ##

# for i in $*; do
#       if [ $i = "the evil profile manager" ]; then
#             echo FALSE
#             exit 0
#       fi
# done


echo TRUE
exit 0
```

5. Make your changes to the scripts, then replace the method:

   ```
   wputpolm -v TaskLibrary NetViewTasks tl_val_man_nodes < /u/lynn/val_nodes

   wputpolm -v TaskLibrary NetViewTasks tl_val_prof_mgrs < /u/lynn/val_mgrs
   ```

6. In the policy region NetView, set the validation policy for TaskLibrary to NetViewTasks.

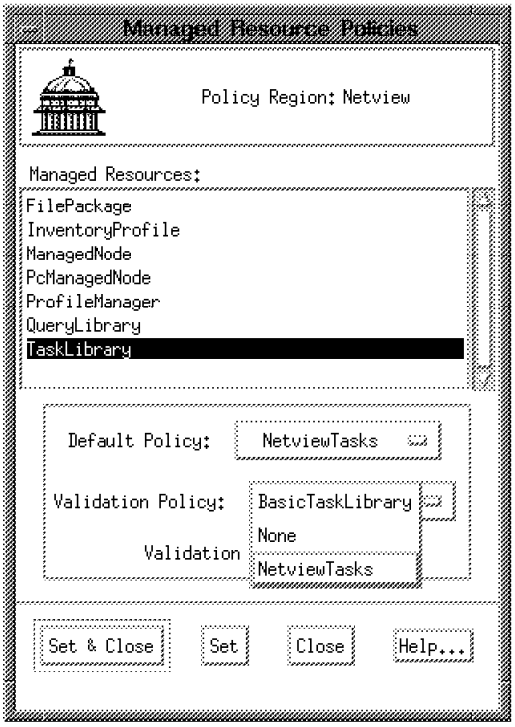


Figure 463. Selecting Validation Policy

The policy will take effect for tasks and jobs in the NetView policy region.

If you drag and drop the task over a resource that is not permitted by the validation policy, you will see the following window:

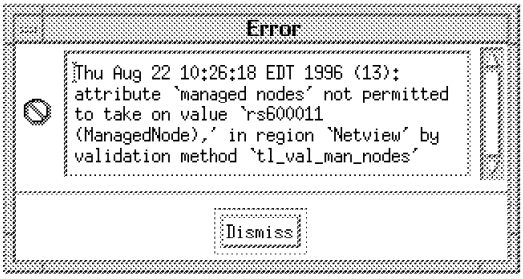


Figure 464. Validation Error Message

If you run wruntask from the command line, you will see this:

```
Thu Aug 22 10:28:55 EDT 1996 (13): attribute `managed nodes' not
permitted to take on value `rs600011 (ManagedNode),' in region `NetView'
by validation method tl_val_man_nodes'
```

Note: See the comments in the *Tivoli Management Platform User's Guide* about limitations in enforcing validation policies when using drag and drop.

15.3 Using the Tivoli Application Extension Facility (AEF)

The *Tivoli/Admin User and Group Management Guide* shows an example of using Tivoli/AEF to add new properties to the user profile and setting up the user interface to input values for those properties. However, the new properties that are added are only information fields and are not used in the actual definition of the user at the endpoint.

In this example we take this a little further by showing how AEF can be used to modify TME 10 User Administration to do the following:

- Add a field to the user profile dialogs that will set the Login AUTHENTICATION GRAMMAR.
- Set this field to a specified value when users are created on AIX 4 systems.
- Change the value at the endpoints when this field is modified in the user profile.

Refer to the *Tivoli/AEF User's Guide* and in particular Chapter 10 *Tivoli/AEF and Profiles* for additional information.

First we need to try to get a picture of how this can all fit together:

1. When a new UNIX user is added to a user profile, there are a number of properties that can be set to define how the user will be set up on the UNIX endpoints.

 he property names appear in the column headings on the User Profile Properties window.

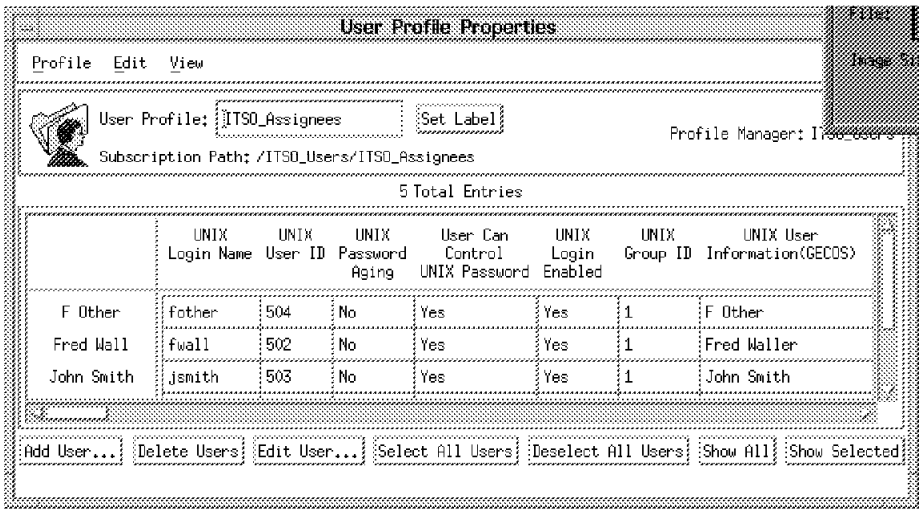


Figure 465. Column Headings Show the Names of the User Profile Properties

The property names can also be seen by selecting **Edit -> Default Policies** from the User Profile Properties menu bar.

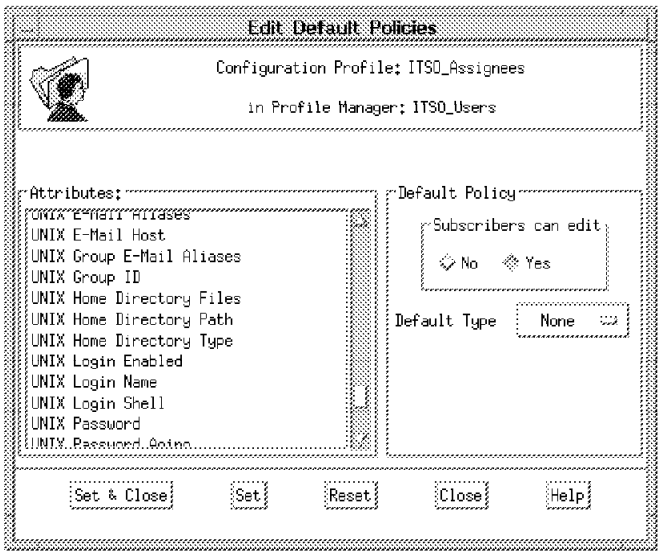


Figure 466. The List of Attributes Equate to the User Profile Properties

Most of the properties that you would want to define for a user ID in any of the managed node environments are listed in the default profiles. However, there are certain items that are not covered. One such property, which is unique to AIX Version 4, is the Authentication Grammar field. By setting this field to DCE it is possible to use DCE authentication in place of normal UNIX password authentication when logging in to an AIX V4 system. We want to add this as a new property that will be added as a column into the panel displayed in Figure 465 on page 479 and as a new attribute in Figure 466.

To add the new property to the User Profile ITSO_Assignees:

```
waddprop @UserProfile:ITSO_Assignees AIX4_Authentication compat
```

This will add the property with a default value of compat and the property will appear with a column heading of AIX4_Authentication.

If you want the property on all user profiles, run the waddprop command for each one.

The name of the property AIX4_Authentication is the name you use later in the definition for this field in the panel. It is the name field that links the panel and any changes made to the value, to the actual property itself.

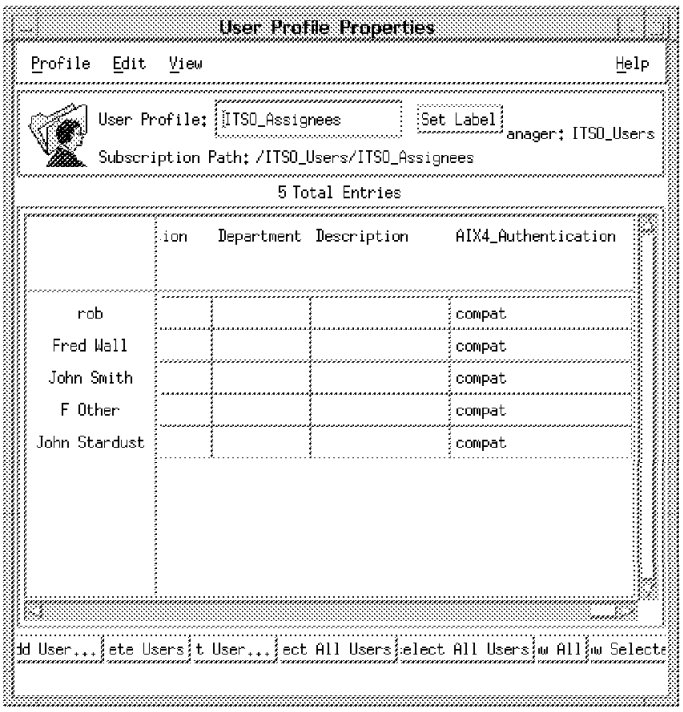


Figure 467. AIX4_Authentication Column Added with Value of Compat

2. The example in the *Tivoli/Admin User and Group Management Guide* suggests adding a default policy for the new property. However, we found it was already set; presumably the waddprop command also defined a default policy, setting the constant value to compat.

 If you find there is no default policy set, you can do the following:

   ```
   wputpol -d -c compat @UserProfile:ITSO_Assignees AIX4_Authentication
   ```

3. We are adding a property that is relevant only to AIX 4, not to every UNIX system, so it can be useful to locate the property in a new category or subcategory. separate from the standard UNIX properties. This makes it easier to pick out the values you may want to change in Add or Edit User.

 Select **Add User** on the User Profile properties window. You will see a Category button with the default set to All.

 This will show the subcategories for all categories.

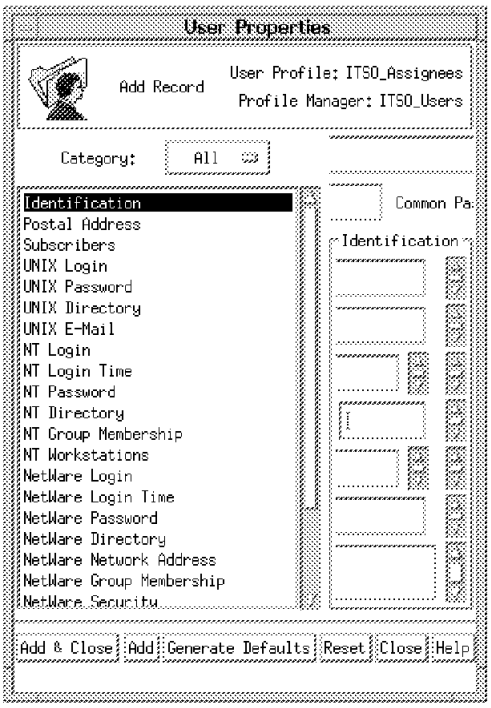


Figure 468. List of Subcategories for All Categories

To filter the list to show subcategories that relate to UNIX properties, click on **Category** and select **UNIX**.

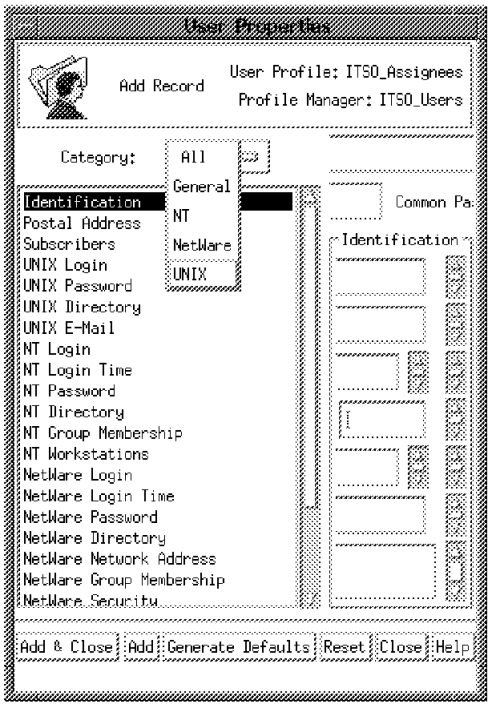


Figure 469. Select UNIX Subcategories

Each subcategory has a panel associated with it, that displays the properties within that subcategory. So the UNIX Login subcategory contains the Login Name, UID, GID, etc. properties.

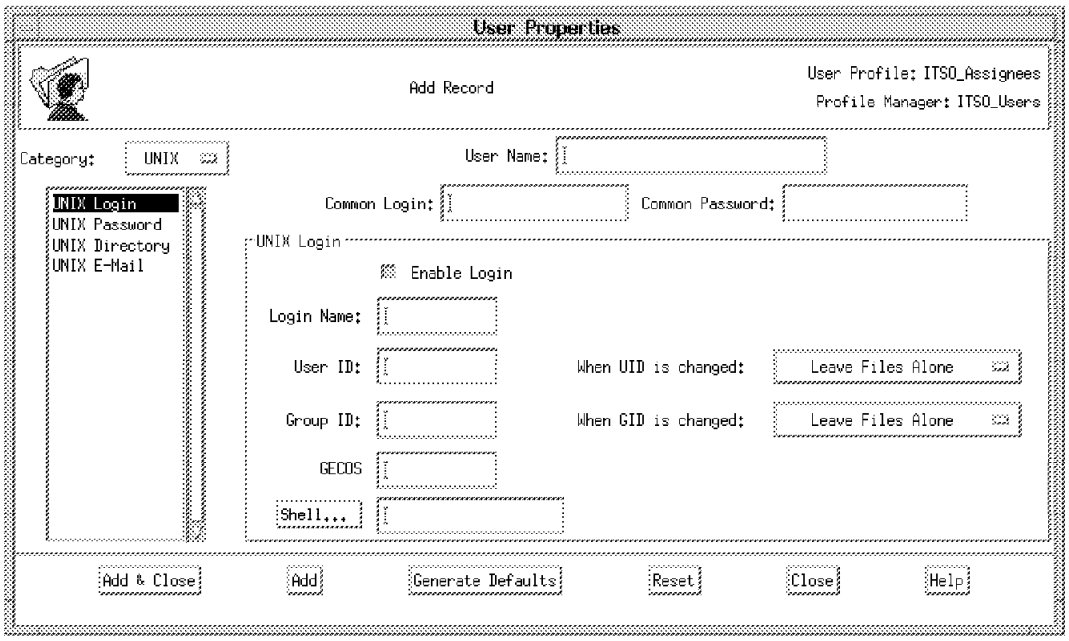


Figure 470. Properties in the UNIX Login Subcategory

We could add a new subcategory called AIX4 into the existing UNIX category, but to expand the example slightly to demonstrate setting up new categories as well as subcategories, we will create a completely new category called AIX4 with a subcategory called Login Authentication.

a. To create the category:

   ```
   wcrtusrcat AIX4
   ```

 We initially called the category UNIX_AIX4 but we noticed that when we then created a subcategory, it appeared to truncate the category name to 8 characters. We didn't find this documented, but we suggest you keep the category name to a maximum of 8 characters.

b. To create the subcategory:

   ```
   wcrtusrsubcat -m "Login Authentication" -c AIX4 user_auth
   ```

 Make a note of the name you allocate to this subcategory. You have to use the same name in the definition of the panel that will be displayed with this subcategory.

 To list out all subcategories:

   ```
   wlsusrsubcat
   ```

 The output will include:

   ```
   user_auth (Label: Login Authentication, Category: AIX4)
   ```

 If you select **Add User** at this stage, you will see the new category AIX4, but the subcategory will not appear as we have not yet created a panel for the subcategory.

4. To create a panel for this subcategory, we must first decide how we want it to look.

 Look at the layout of the panel associated with the UNIX password subcategory. This shows examples of several different types of layout for the property fields.

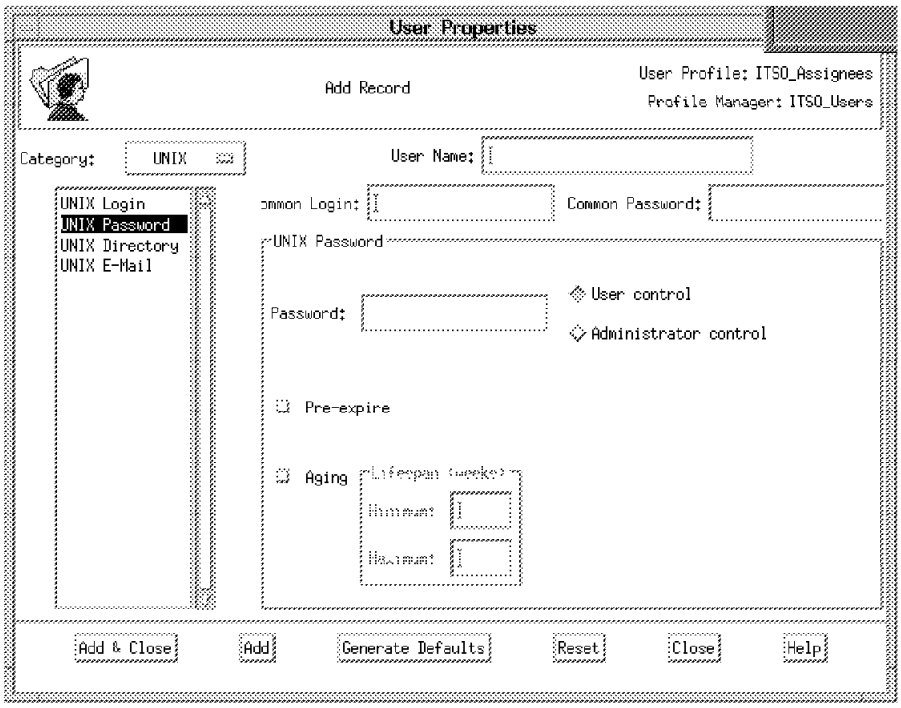


Figure 471. Examples of Different Layouts for Property Fields

The property we want to create has only two options, compat or DCE. So the field on the UNIX Password panel that would seem to match best, is the User control and Administer control choice. Another option might be a scrollable list, although this would not be very relevant where there are only two items to scroll!

To retrieve the code that defines the UNIX Password panel:

List all the dialogs associated with the user interface:

```
wlsdialog -r UserGui
```

Included in a long list, you will find:

```
UDirectoryGroup
UEmailGroup
ULoginGroup
UPasswordGroup
```

These dialogs relate to the four subcategories in the UNIX category, so we can retrieve UPasswordGroup.

a. To retrieve the dialog:

```
wgetdialog -r UserGui UPasswordGroup > /tmp/UPasswordGroup.d
```

b. The file that is retrieved needs to be reverse compiled:

```
rdsl /tmp/Upasswordgroup.d > /tmp/Upasswordgroup.dsl
```

c. You can now view /tmp/Upasswordgroup.dsl to see how the UNIX Password panel was defined.

Refer to *Appendix B, CLI Reference in the Tivoli/AEF User's Guide* for detailed explanations of all the commands.

Use the sample dialog in the *Tivoli/Admin User and Group Management Guide* in conjunction with the code retrieved for the UNIX Password panel, to build your own panel for the Login Authentication panel.

You may end up with a file like this:

```
 Partial Dialog
{
    Group
    {
         Border = YES;
         Layout = VERTICAL;
         Name = user_auth;
         Title = "AIX4 Login Authentication";
         TitlePos = TOP;
         Visible = YES;
         GridHorizontal = 0;
         GridVertical = 0;
         ChildColumnAlignment = STRETCH;
         ChildRowAlignment = STRETCH;

        Group
        {
            Layout = HORIZONTAL;
            Name = authgroup;
            Border = YES;
            ChildColumnAlignment = LEFT;
            ChildRowAlignment = TOP;

            Choice
            {
                  Choices =
                      "compat"{compat},
                      "DCE"{DCE};
                  Layout = HORIZONTAL;
                  Name = AIX4_Authentication;
                  Show = ALL;
                  Sort = NO;
                  ChildColumnAlignment = LEFT;
                  ChildRowAlignment = CENTER;
            }
        }
    }
}
```

Notice the value allocated to Name in the first Group definition. This value must be the same as the name allocated to the subcategory with which this panel will be associated, in this case user_auth.

Also make a note of the value allocated to Name in the Choice definition. This must be the same as the name allocated to the property you added earlier.

d. Compile the code:

```
dsl /tmp/user_auth.dsl > /tmp/user_auth.d
```

Here are some common errors that you might see:

- Ending line 18 with a colon (:) instead of a semicolon (;), results in the message:

```
user_auth.dsl:18:
syntax error
```

- Entering `name = TopGroup` instead of `Name = TopGroup` results in message:

```
user_auth.dsl:19:
name: Unrecognizable Attribute Name specified.
```

- Missing a **{** or **}** anywhere in the code, may result in a message that points to the last line in the code. You will then have to go through the code to match up **{}**.

e. When the code has compiled cleanly, you can preview the panel you have created:

```
tivoli -preview /tmp/user_auth.d
```

This will not start Tivoli, but will just show you the panel you have defined.



Figure 472. Preview New Panel

In our example, we are only adding one property into the subcategory login authentication, but we may want to create other subcategories containing several properties that can logically be grouped together on one panel.

The following is some code that defines two more fields on the panel and changes the layout to get a better look.

```
 Partial Dialog
{
    Group
    {
         Border = YES;
         Layout = HORIZONTAL;
         Name = user_auth;
         Title = "AIX4 Login Authentication";
         TitlePos = TOP;
         Visible = YES;
         GridHorizontal = 0;
         GridVertical = 0;
         ChildColumnAlignment = STRETCH;
         ChildRowAlignment = STRETCH;

        Group
        {
            Layout = HORIZONTAL;
            Name = authgroup1;
            ChildColumnAlignment = LEFT;
            ChildRowAlignment = TOP;

            Choice
            {
                  Choices =
                      "compat"{compat},
                      "DCE"{DCE};
                  Title = "Login Values : ";
                  Border = YES;
                  Layout = HORIZONTAL;
                  Name = AIX4_Authentication;
                  Show = ALL;
                  Sort = NO;
                  ChildColumnAlignment = LEFT;
                  ChildRowAlignment = CENTER;
            }
        }
        Group
        {
            Layout = VERTICAL;
            Name = authgroup2;
            ChildColumnAlignment = LEFT;
            ChildRowAlignment = TOP;

            Text
            {
                  Columns = 10;
                  Title = "Text option 1";
                  Name = login_auth2;
                  ChildColumnAlignment = LEFT;
                  ChildRowAlignment = CENTER;
            }
            Text
            {
                  Columns = 10;
                  Title = "Text option 2";
                  Name = login_auth3;
                  ChildColumnAlignment = LEFT;
                  ChildRowAlignment = CENTER;
            }
        }
    }
}
```

Here we have two groups authgroup1 and authgroup2 within the main group user_auth.

The main group's layout is set to HORIZONTAL, so that the contents of the groups below will be laid out side by side. The group authgroup1 is also HORIZONTAL, but the authgroup2 is VERTICAL, so that the two text fields appear vertically.

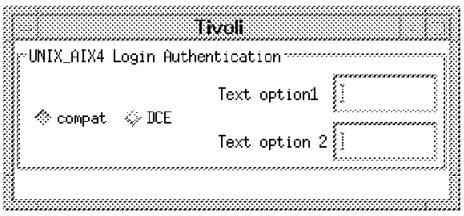


Figure 473. Panel Defined with Extra Fields

Manipulating the layout in this way can give you a panel that is much easier to understand.

It is quite possible that in the two examples above, we have included some code that is unnecessary. However copying existing code and modifying it to suit your needs can sometimes be easier than trying to start each dialog from scratch. As you (and we) become more familiar with the code required to define these panels, you should be able to spot more easily what is and is not required.

At this stage, experiment with the layout, with choice, text, page and switch gadgets to see the effects you can get.

f. When you are happy with the panel you have created, you need to save the dialog and associate it with the subcategory login authentication:

```
wputdialog -r UserGui user_auth < /tmp/user_auth.d
```

You will see the message:

```
Adding resource-wide customization for dialog user_auth and resource
UserGui.
```

The association with the subcategory login authentication is made by the name user_auth. This was the name we also used in the wcrtusrsubcat command and the name in the dsl file:

```
wcrtusrsubcat -m "Login Authentication" -c AIX4 user_auth

wputdialog -r UserGui user_auth < /tmp/user_auth.d

dsl file :
 Partial Dialog
{
    Group
    {
         Border = YES;
         Layout = HORIZONTAL;
         Name = user_auth;
```

You will probably need to restart your desktop to pick up and display the panel.

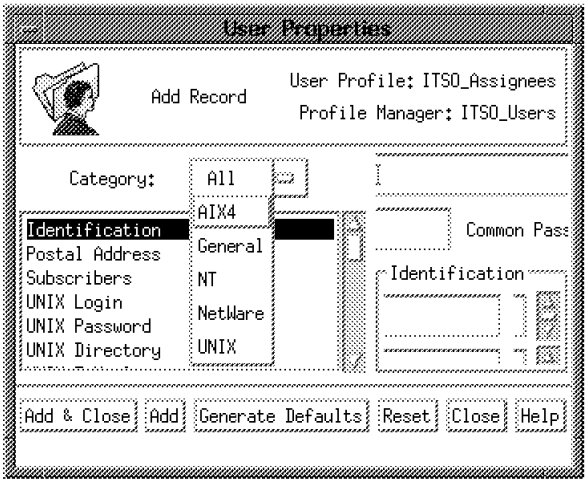


Figure 474. AIX4 Category Now Available

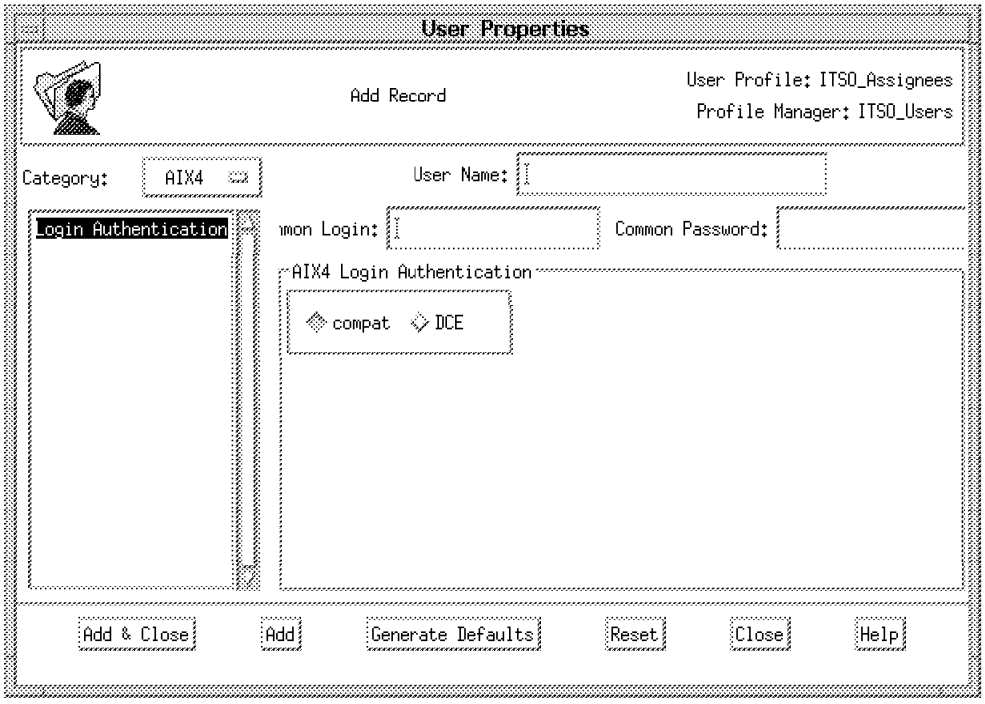


Figure 475. AIX4 Login Authentication Panel

The new category, subcategory and panel will be available from all the user profiles in ITSO_Users, but will only appear on the User Profile Properties panel if you have added the property to that user profile with the waddprop command.

If you find you can change the Login Authentication panel, but the value does not change on the User Profile Properties window, check that the name you have set in the gadget definition in the dsl file matches exactly the name you have allocated to the property, that is, AIX4_Authentication.

You now have the property displayed on the User Profile Properties window and you should be able to edit a user and change the value in the Login Authentication field.

If you want to add a property to an existing panel:

1. Retrieve an existing dialog:

   ```
   wgetdialog -r UserGui UPasswordGroup > /tmp/UPasswordGroup.d

   rdsl /tmp/UPasswordGroup.d > /tmp/UPasswordGroup.dsl
   ```

2. Add the definition of the property into the file.
3. Compile the code:

   ```
   dsl /tmp/UPasswordGroup.dsl > /tmp/UPasswordGroup.d
   ```

4. Put the customized dialog back into UserGui:

   ```
   wputdialog -r UserGui UPasswordGroup < /tmp/UPasswordGroup.d
   ```

 The UNIX Password panel will now include the Login Authentication field.

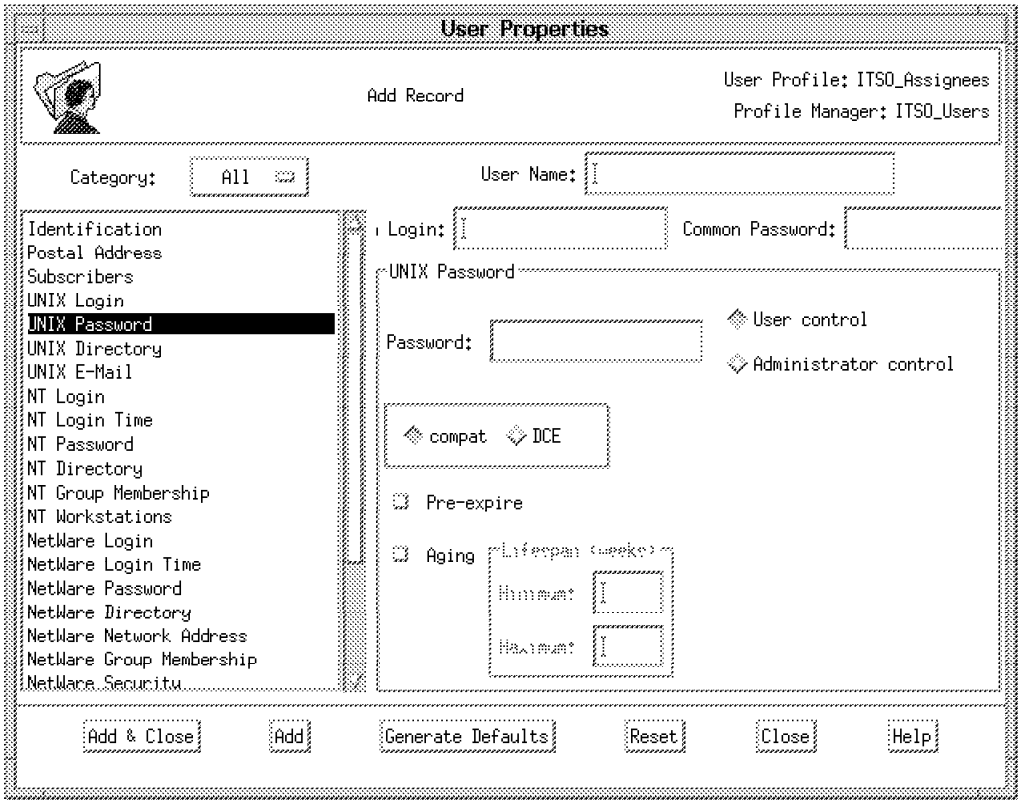


Figure 476. UNIX Password Panel Including the Login Authentication Field

5. If you want to remove the customized version and return to the standard panel:

   ```
   wrmdialog -r UserGui UPasswordGroup
   ```

 This removes only the customized version leaving the original intact.

The next step is to use our new property in the creation and change of user Ids on AIX4 endpoints, so that a user will have the SYSTEM attribute set to a default of compat or to DCE.

When Tivoli/Admin creates a user, it will use the values set in the user profile which were either set from default polices or set by the administrator. All other attributes that a user can have will be set from the defaults of the particular

platform where the user is being created. On an AIX4 system, the default login authentication is SYSTEM=compat. If we want to create or change a user to SYSTEM=DCE we need a way of passing this value from the profile to the endpoint.

Here is a way to achieve this:

1. Profile-based applications such as Tivoli/Admin support a mechanism for invoking *actions* when profiles are distributed. These actions are shell scripts that are associated with creation, deletion or modification of profile records.

 The actions can be set to run at different stages of the distribution:

 - Run before the profile is distributed.
 - Run on the host where the profile resides after the profile is distributed to its subscribers.
 - Run on each endpoint before the profile is distributed.
 - Run on each endpoint after the profile is distributed.

 In our example, we want to run a script on an endpoint after the profile is distributed.

 a. The user will be created as normal with the values passed from the standard properties in the user profile and with platform defaults for all other attributes.

 b. When the user has been created, we will execute a script to change the SYSTEM attribute to a value passed to the script from the new property we have included in the user profile.

 The script will do the following:

 - Check that the values being passed to it are valid
 - Check the operating system level to see if we need to continue
 - Check that the user exists
 - Change the SYSTEM attribute to the value passed to the script

```
#!/bin/sh
# Change login Authentication value
# valid values are : DCE/compat
#
#set -xv
#exec > /tmp/login_auth.debug 2>&1
MY_NAME=login_authentication.sh
CLI_EXIT_OK=0
CLI_EXIT_USAGE=1
CLI_EXIT_ERROR=1
USERNAME=$1
AUTH_VALUE=$2

case $AUTH_VALUE in
   DCE compat ) :
                ;;
            * ) echo "$MY_NAME : $AUTH_VALUE is an invalid value" >&2
                exit $CLI_EXIT_ERROR
                ;;
esac

PATH=/usr/bin:/usr/ucb:$PATH; export PATH

# Check number of arguments
if [ $# -ne 2 ]; then
   echo "Usage: $MY_NAME username auth_value" >&2
   exit $CLI_EXIT_USAGE
fi

os=`uname -sv`
echo $os   grep "AIX 4" > /dev/null

if [ $? -eq 0 ]; then
  lsuser $USERNAME
  if [ $? -ne 0 ]; then
    echo "$MY_NAME: user $USERNAME does not exist" >&2
    exit $CLI_EXIT_ERROR
  fi

  chuser SYSTEM=$AUTH_VALUE $USERNAME
  if [ $? -ne 0 ]; then
    echo "$MY_NAME: unable to change Login authentication" >&2
    exit $CLI_EXIT_ERROR
  fi
fi

exit $CLI_EXIT_OK
```

Figure 477. Action Script login_authentication.sh

Test the script manually on an AIX4 system and any other platform where you will be creating users, for example, AIX 3.2 or SunOS.

2. To add the action to the user profile ITSO_Assignees:

   ```
   waddaction -A -c @UserProfile:ITSO_Assignees AIX4_login_auth \
         args='$login_name','$AIX4_Authentication' < login_authentication.sh
   ```

 where:

 - -A specifies run at each endpoint after distribution.
 - -c associates the action with creation of a profile.
 - @UserProfile:ITSO_Assignees is the name of the user profile to which this action applies.
 - AIX4_login_auth is a name allocated to the action. The name does not relate to anything else; it just needs to be unique within this user profile.

- `args='$login_name','$AIX4_Authentication'` are the arguments passed to the script. These arguments can be fixed values:

  ```
  args=TRUE
  ```

 or as in our example, values of properties in the user profile record.

 When passing values of properties, they are expressed as $propertyname and need to be enclosed in quotes. We found several different examples of where the quotes should go, but the example above worked.

 We had a few problems in finding out the exact name of the standard properties. For example $login_name is the correct name (in TME 3.0) for the UNIX login name property, but we could not find a command or documentation that listed them.

 However, as the property name must match the name in the dsl file that defines the gadget, we discovered the name by doing the following:

 a. List the dialogs associated with the user interface:

     ```
     wlsdialog -r UserGui
     ```

 b. As we did earlier for the UPasswordGroup dialog:

     ```
     wgetdialog -r UserGui ULoginGroup > /tmp/ULoginGroup.d

     rdsl /tmp/ULoginGroup.d > /tmp/ULoginGroup.dsl
     ```

 c. View /tmp/ULoginGroup.dsl to find the name for each gadget. This name will match the property name.

 This seems a rather complicated way of getting the information and it is probable there is a command to get the information that we have not found yet!

- < login_authentication.sh is the shell script you want to execute at the endpoints.

If you want this action to apply to all user profiles, run the waddaction command for each one.

To list the actions you have added:

```
wlsactions @UserProfile:ITSO_Assignees
```

This will list all actions, but you can filter the output by specifying:

```
wlsactions [-b   -a   -B   -A ] [ -c   -d   -m] \
@UserProfile:ITSO_Assignees
```

To remove an action:

```
wrmaction -A -c @UserProfile:ITSO_Assignees AIX4_login_auth
```

The remove command must specify the same flags with which it was added.

3. The action script we are using when creating a user actually waits until the user has been created then performs a change. This means that we can use the same script in an action where the user record is modified.

 An action associated with modify operations can be defined on a per-attribute basis. This means that the action will only be triggered if a particular attribute changes rather than executes for every modification.

 The action therefore needs to be associated with a particular attribute and this is achieved by making the action name the same as the profile property (attribute):

```
waddaction -A -m @UserProfile&:ITSO_Assignees AIX4_Authentication \
args='$login_name,'$AIX4_Authentication' < /tmp/login_auth.sh
```

4. To retrieve an action that was created with the waddaction command:

   ```
   wgetaction -A -m @UserProfile&:ITSO_Assignees AIX4_Authentication \
   > /tmp/AIX4_Authentication.sh
   ```

 This command will retrieve the script into the named file and will also list the arguments that you specified:

   ```
   script arguments:  $login_name  $AIX4_Authentication
   ```

 We tested this customization of Tivoli/Admin user profiles against only AIX systems. If your managed nodes subscribing to a user profile include NT and NetWare systems, we are not sure what effect trying to execute this action might have.

Chapter 16. Integrating NetView for AIX with Other TME Functions

TME 10 NetView (formerly NetView for AIX) is the most significant application from the former SystemView for AIX that has become one of the TME 10 products. NetView for AIX will be more tightly integrated into the TME framework as time goes by, but at the time of writing this document, integration was limited to the ability of passing events from NetView for AIX to the TME 10 Enterprise Console (T/EC).

In this chapter we show some examples of customization and sample user code that allows NetView for AIX to operate with the TME Desktop and T/EC console in a more integrated way than only through T/EC. The examples are as follows:

- An example of how the NetView for AIX GUI can be customized to replace the normal NetView for AIX event application with a T/EC console.
- Some basic examples of NetView for AIX rulesets that can be used to pass events on to the T/EC.
- An example that allows TME tasks to be executed from NetView for AIX submaps.
- Sample code that allows you to create NetView for AIX collections based on TME policy regions and profile managers.

16.1 Customizing the NetView for AIX Environment

We have seen in Chapter 11, “Tivoli/Enterprise Console Adapters” on page 355 that there are two ways to forward NetView for AIX events to the T/EC console. A logical outcome of this is that you may not want to run the NetView for AIX events display at all. The T/EC event console is a more flexible and powerful way to distribute required information to multiple distributed people and places. It also takes a more varied range of event message inputs. If you do suppress the NetView events display and replace it with a more heavily filtered T/EC display you do not lose anything; the events are still logged in NetView so you can review them later.

The NetView for AIX event display application is a program called nvevents. It starts within the *control desk* portion of the main NetView for AIX window. You can prevent this from starting up when NetView is started by modifying the registration file, as follows:

1. Change directory to /usr/OV/registration/C/ovsnmp. on the NetView for AIX system.
2. Edit the nvevents file and remove the -Initial flag. Figure 478 on page 496 shows the resulting file. The point at which the -Initial flag was removed is marked **1**.

© Copyright IBM Corp. 1996

```
Application "Event Display Application"
{
    Description {
        "SNMP Event Notification Viewer"
    }

    Version "V4R1";

    Copyright {
      "Licensed Program Product:",
      "  NetView for AIX",
      "(C) COPYRIGHT International Business Machines Corp. 1992,1994",
      "(C) COPYRIGHT Hewlett-Packard Co. 1992",
      "    All Rights Reserved",
      "US Government Users Restricted Rights - Use, duplication,",
      "or disclosure restricted by GSA ADP Schedule Contract with",
      "IBM Corp. and its licensors.",
      ""
}

    /*
     * Use -Shared so that only one copy of application is run.
     * If user selects menu item again, OVwAction causes the existing window
     * to raise to top of stack.
     *
     * Use -Initial so that the events categories window gets displayed when
     * OVW starts up.
     *
     */
1   Command -Shared "${nvevents:-/usr/OV/bin/nvevents}" ;

    MenuBar "Monitor"
    {
        <100> "Events"          _E  f.menu "Events";
    }

    Menu "Events"
    {
      <100>     "Current Events..."               _S  f.action "events";
    }
/******* Popups ***/

Objectmenu
    {
        <100>   "Edit"          _E              f.menu P_Edit;
        <100>   "View"          _V              f.menu P_View;
        <100>   "Options"       _p              f.menu P_Options;
        <100>   "Monitor"       _M              f.menu P_Monitor;
        <100>   "Test"          _T              f.menu P_Test;
        <100>   "Tools"         _o              f.menu P_Tools;
        <100>   "Administer"    _A              f.menu P_Administer;
    }

/**********/

    Tool "Events"
    {
        Icon Gif "/usr/OV/icons/gifs/dynamic_events.gif" ;
        DragBitmap "/usr/OV/icons/drag-bitmaps/dynamic_events.xbm";
        LabelColor "black";
        SelectionMechanism double-click, drag-drop ;
        Action "events";
    }
/********/
    Menu "P_Monitor"
    {
        <100> "Events"          _E  f.menu "P_Events";
    }
```

Figure 478 (Part 1 of 2). Registration File for nvevents

```
    Menu "P_Events"
    {
      <100>     "Current Events..."               _S  f.action "events";
    }
/*******/
    Action "events" {
        SelectionRule isNode    isVertex    isGraph;
        MinSelected 0;
    }
    Action "nveventsApi" {
        MinSelected 0;
    }
}
```

Figure 478 (Part 2 of 2). Registration File for nvevents

16.2 Configuring NetView for AIX Rulesets

As we discussed in 11.5, “The NetView for AIX Ruleset Event Adapter” on page 370 the T/EC adapter introduced by PTF U443133 is the easiest way to select NetView events for delivery to the T/EC. In this section we show how to build a simple NetView ruleset for the adapter.

In this example, we make a ruleset that forwards node down and node up events to the T/EC event server. If you want to know more about designing NetView rulesets you may want to refer to *Examples Using NetView for AIX Version 4*, SG24-4515. The rulesets are stored in directory /usr/OV/conf/rulesets. This directory has 777 permission so that any user can make new rulesets (but cannot change/modify other users′ rulesets).

To create the ruleset:

1. Start Ruleset Editor by either entering `/usr/OV/bin/nvrsEdit` from the command line or by clicking on the icon in the NetView for AIX Tools palette or by selecting **Tools** and then **Ruleset Editor** from the menu bar. The two windows shown in Figure 479 on page 498 are displayed.

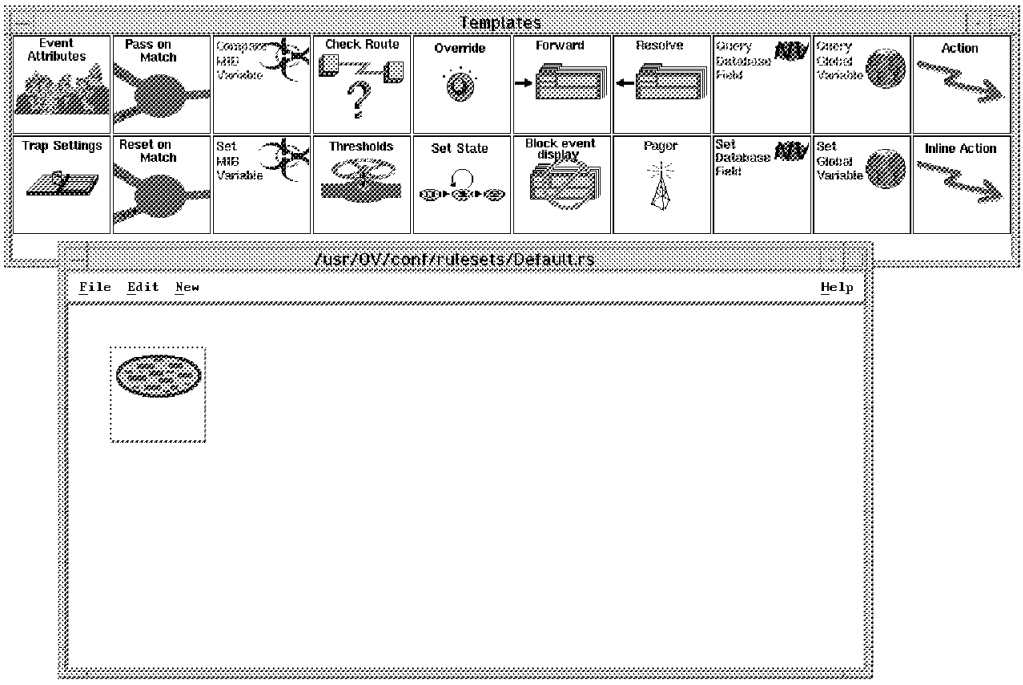


Figure 479. Ruleset Editor

2. Select **File->New**. The window title changes from /usr/OV/conf/rulesets/default.rs to * Untitled *.
3. Drag the Trap Settings icon from the Templates window to the Ruleset Work Area. Then, another window comes up. Specify the Node Down event and select **OK** on the Trap Settings window. Fill in the following:

 Enterprise Name netView6000 1.3.6.1.4.1.2.6.3

 Event Name IBM_NVNDWN_EV

 Specific Specific 58916865

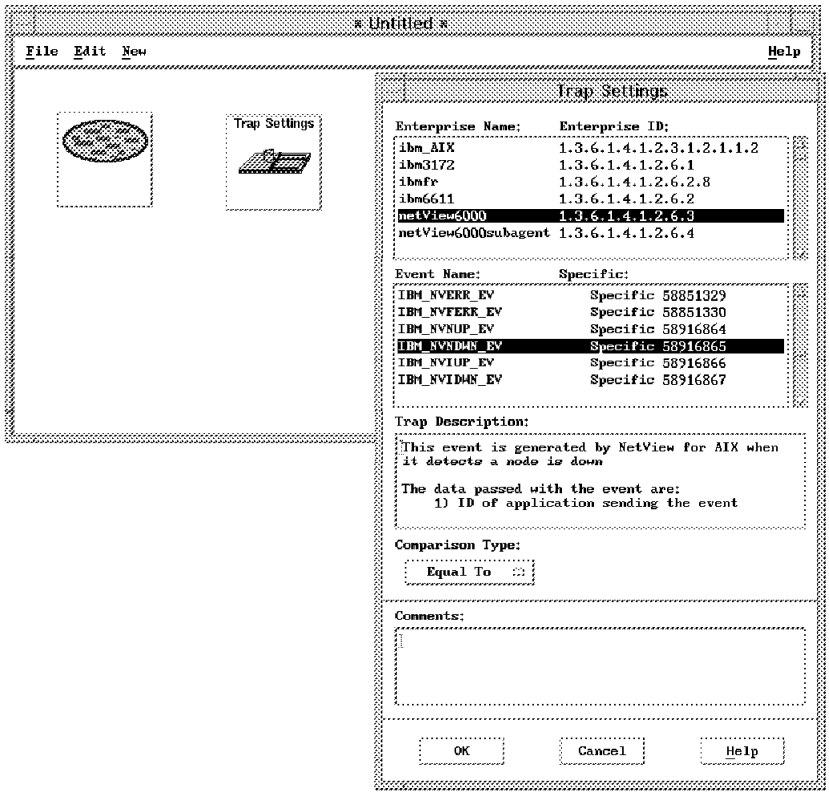


Figure 480. Node Down Trap Setting

4. Connect the event stream node to the trap settings node. Select **Edit->Connect Two Nodes** from the menu bar. Your mouse pointer becomes a connection icon. Click the **Event Stream node** icon first, then the **Trap Settings** icon next. Now, those two nodes are connected.

5. We will also forward the node up event to the T/EC server. Do steps 2 and 3, with the following trap settings:

 Enterprise Name netView6000 1.3.6.1.4.1.2.6.3

 Event Name IBM_NVNUP_EV

 Specific Specific 58916864

6. Next, pass the selected events to the Event Display application by dragging **Forward** node from the Template into the Ruleset Work Area and connect it to the Node Down Trap Setting node. Do the same operation for the Node Up Trap Setting node.

7. Now, your ruleset looks as shown in Figure 481 on page 500.

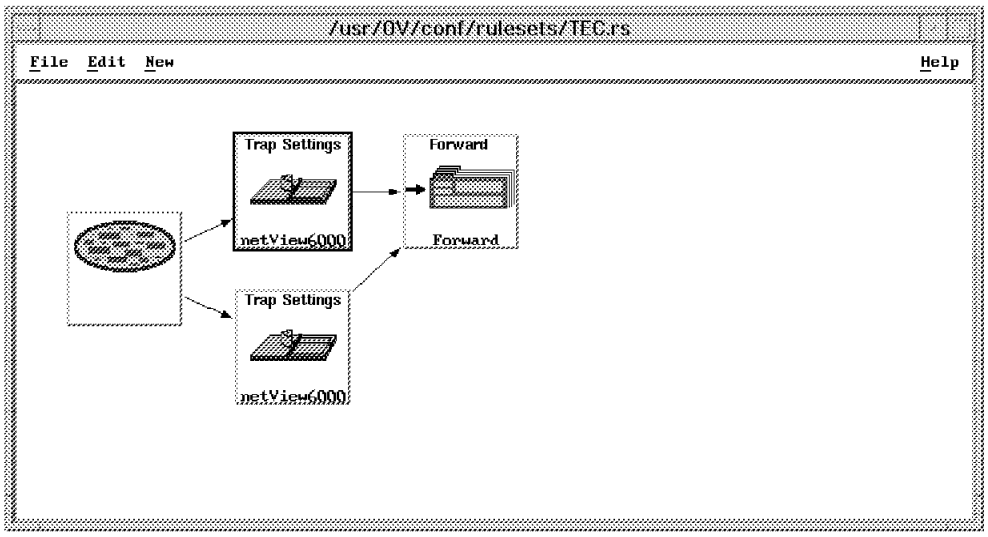


Figure 481. Completed Example Ruleset

Save your rule base for the later operation. Select **File -> Save As...** from the menu bar. Save, creating the ruleset as TEC.rs, and exit the Ruleset Editor. By this action, your created ruleset is recognized by NetView for AIX.

16.2.1 Defining Ruleset Working with Tivoli

NetView for AIX PTF U443133 allows forwarding NetView for AIX events to T/EC by allowing the NetView for AIX nvserverd to work as a T/EC event adapter. To enable this new function, we need to define which NetView for AIX ruleset is used to forward events to T/EC server. This is quite easy. We can do it through SMIT.

In this example, the T/EC server is rs600020, and the NetView for AIX ruleset is TEC.rs which we made in 16.2, “Configuring NetView for AIX Rulesets” on page 497. The following steps enable the new PTF support:

1. You need to be a root user. If so, enter `smitty nv6config` from the command line.
2. Select **Configure for Tivoli**.

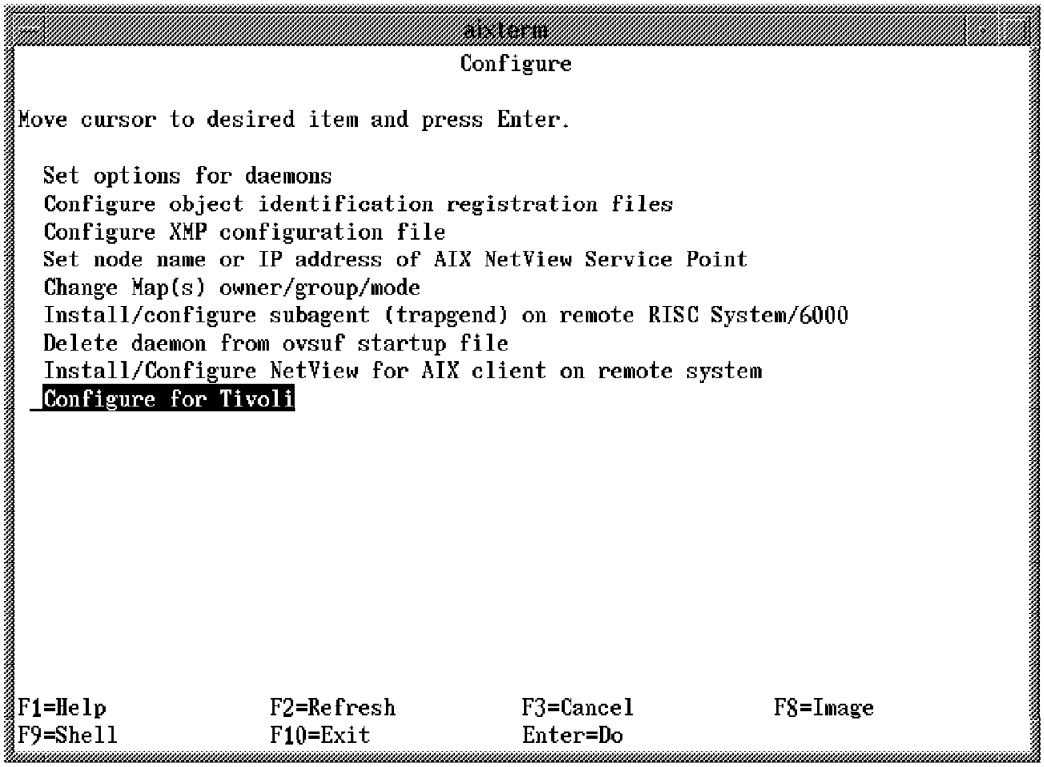


Figure 482. NetView for AIX Configuration Panel

3. We want to forward Node Down/Up events to T/EC server, so change the Forward events to Tivoli event server? field to yes.
4. Type rs600020 into the Tivoli event server host name field.
5. For the NetView rule name, list all rulesets in your machine by pressing PF4, and select **TEC.rs**. Then press Enter.

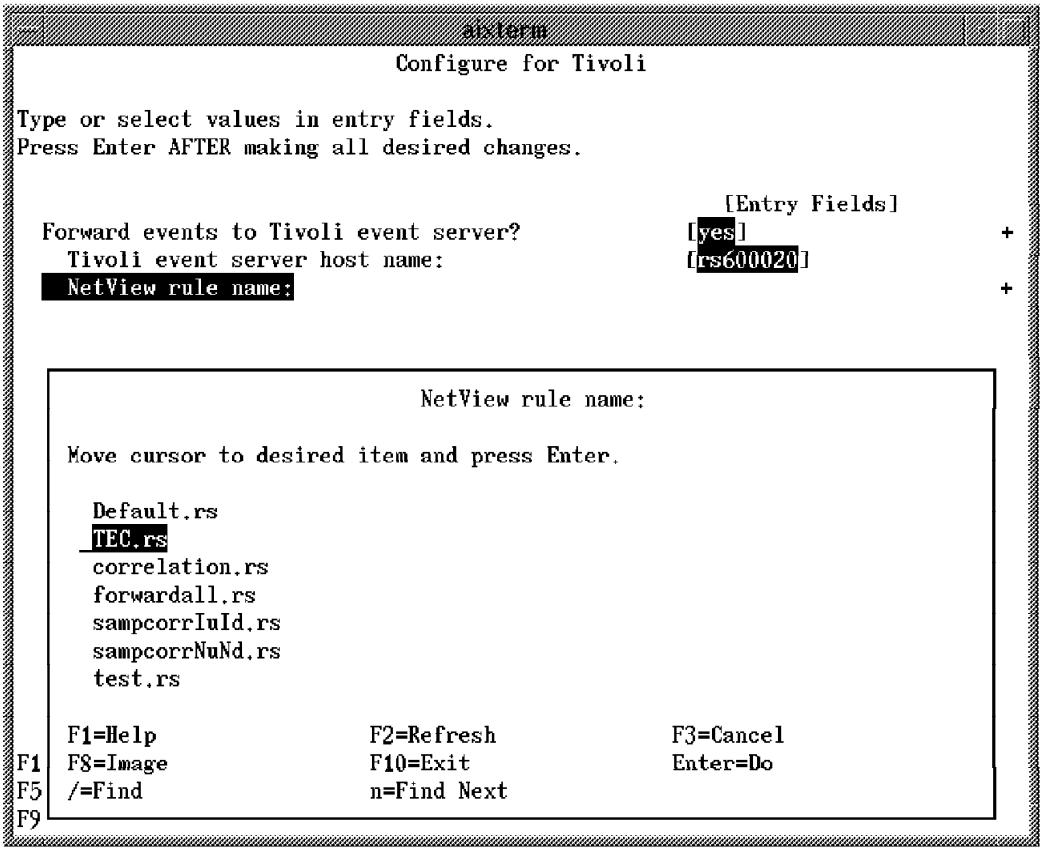


Figure 483. Configuration for Tivoli Panel

This operation's result is stored in /usr/OV/conf/tecint.conf. You can confirm what ruleset is used with the T/EC server by checking this file.

16.3 NetView for AIX Menu Integration

In this section we show how tasks set up in TME can be integrated into NetView for AIX menus, to execute against objects selected from the NetView for AIX map. This is an example of synergy between NetView for AIX and the TME environment: TME provides a facility to execute commands on one or more distributed systems in a secure, reliable way. NetView provides a convenient way to locate and select systems from a graphical map.

To achieve this we do the following:

1. Define NetView for AIX registration files to provide TME menu items.
2. Write scripts to execute TME tasks from the TME menu items.

The tasks to be integrated are as follows:

- Tasks, as discussed earlier in Chapter 4, "Task Libraries, Tasks and Jobs" on page 81.
 - Show_Filesystems
 - Set_isTME_NODE
- An additional task we create:
 - Who

This task is be placed in General_Tasks and executes the /usr/bin/who command to list the users who are logged in to the systems.

In our environment, we had an HPUX and Solaris 2.3 that we wanted to include in our selected task endpoints. We created the task with the platforms supported as follows:

```
Generic - /usr/bin/who from rs600011 (AIX)
PA-RISC / HPUX 9 - /bin/who from machine hp
SPARC / Solaris - /usr/bin/who from machine sun
```

16.3.1.1 NetView for AIX Registration Files

The menu items displayed on the top line of the NetView for AIX interface are defined in registration files located in files and directories below the /usr/OV/registration/C directory. We do not touch any of the standard files supplied with NetView for AIX; we create our own as follows:

1. Make a new subdirectory under the NetView registration directory, with command `mkdir /usr/OV/registration/C/TME_menus`
2. Create a file in the directory called ″General_Tasks″.

```
Application ″TME - General Tasks″ {
   Description {
     ″Simple TME tasks″
   }
   MenuBar ″TME″  {
           ″General Tasks″     f.menu  ″TME_simple_tasks″;
   }
        Menu ″TME_simple_tasks″ {
           ″Show Filesystems ″     f.action        ″TME_showfs″;
           ″Show Filesystems ″     f.action        ″TME_showfs_coll″;
           ″Who is Logged in ?″    f.action        ″TME_who″;
           ″Who is Logged in ?″    f.action        ″TME_who_coll″;
   }
        Action ″TME_showfs″ {
           SelectionRule isTME_NODE;
           MinSelected 1;
           Command ″/u/lynn/general_tasks.sh Show_Filesystems″;
   }
        Action ″TME_showfs_coll″ {
           SelectionRule isTME_Collection;
           MinSelected 1;
           MaxSelected 1;
           Command ″/u/lynn/general_tasks.sh Show_Filesystems″;
   }
        Action ″TME_who″ {
           SelectionRule isTME_NODE;
           MinSelected 1;
           Command ″/u/lynn/general_tasks.sh Who″;
   }
        Action ″TME_who_coll″ {
           SelectionRule isTME_Collection;
           MinSelected 1;
           MaxSelected 1;
           Command ″/u/lynn/general_tasks.sh Who″;
   }
}
```

Figure 484. General_tasks Registration File

3. Create a file called ″NetView_Tasks″

```
Application "TME - NetView Tasks" {

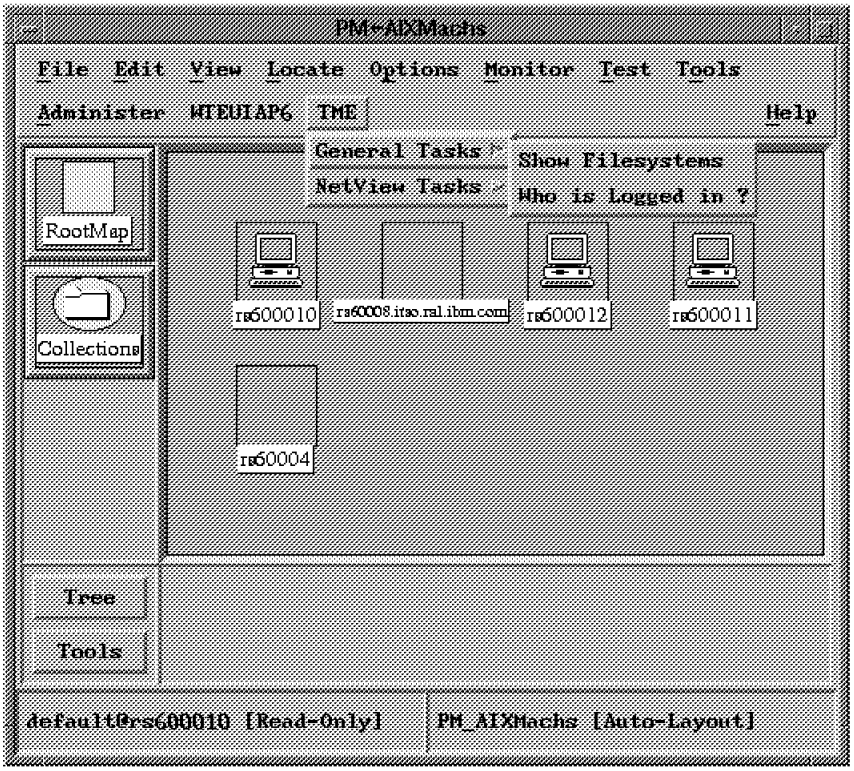
   Description { TME Examples - "Netview TME tasks"
   }

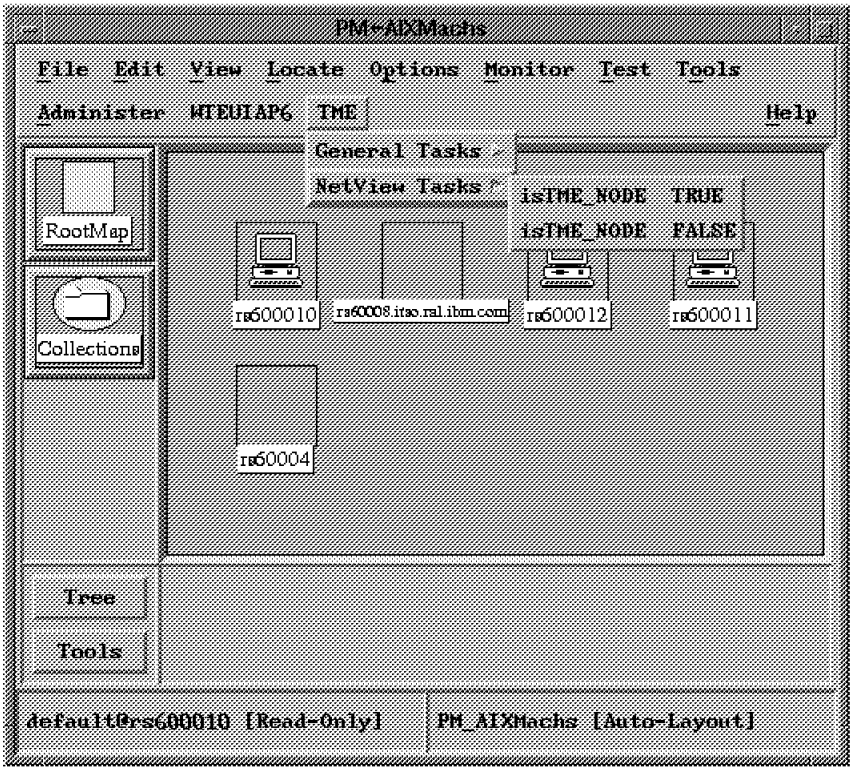
   MenuBar "TME"  {
           "NetView Tasks"     f.menu  "TME_netview_tasks";
   }
        Menu "TME_netview_tasks" {
           "isTME_NODE  TRUE"     f.action        "isTME_NODE_TRUE";
           "isTME_NODE  FALSE"     f.action       "isTME_NODE_FALSE";
   }
   Action "isTME_NODE_TRUE" {
           MinSelected 1;
           Command "/u/lynn/Netview_tasks.sh TRUE";
   }
   Action "isTME_NODE_FALSE" {
           MinSelected 1;
           Command "/u/lynn/Netview_tasks.sh FALSE";
   }
}
```

Figure 485. NetView_tasks Registration File

Refer to *NetView for AIX Installation and Configuration*, SC31-8163 for full details on the contents and format of registration files.

The following menu items will now appear on your NetView for AIX window:





16.3.1.2 Write the Scripts to Run the TME Tasks

When creating scripts to execute TME tasks from NetView for AIX, do the following:

- Keep the number of scripts to a minimum. This will make managing and maintaining them much easier.
- Use variables wherever possible to make the scripts generic.

The two tasks running under General Tasks are simple tasks that require only the task name to be passed. The NetView for AIX registration file executes a single script for both options and passes the name of the task to the script.

1. Create a script file called general_tasks.sh.

```
#!/bin/ksh
#####################################################################
#####################################################################
##
##  Filename: general_tasks.sh
##
##  Date of the last modification: 08/06/96
##
##  Description: Generic script to run simple tasks on hosts
##               or Collection selected from the NetView map.
##
##  Arguments: <task_name>
##
##
#####################################################################
#####################################################################

# Set the task name from the parameter passed to the script

task_name=$1


# set up Tivoli environment

. /etc/Tivoli/setup_env.sh

# Check if a Collection rather than hosts was selected from the Map.
# Collections will always have a name ending in .CF
# $OVwSelections is the NetView variable that contains the name(s) of
# selected objects.

collection=`echo $OVwSelections   grep ".CF"`

# If we are executing the task against specific hosts, the flag that is passed
# to the wruntask command is "-h"; if executing against a collection (which
# equates to a Profile Manager), the flag is "-p"

for i in $OVwSelections
 do
   if [[ -z "$collection" ]]; then
     flag="-h"
     host=`echo $i   cut -d "." -f1`
     target="$target $host"
   else
     flag="-p"
     profile=`echo $i   cut -d "." -f1   sed -e 's/PM_//'`
     target="$target $profile"
   fi
 done

# Use the xnmappmon command to display the output from the task in a
# scrollable window.

xnmappmon  -commandTitle "$task_name" -cmd /u/lynn/run_general_tasks.sh \
 $task_name $flag "$target"

exit
```

Figure 486. general_tasks.sh Shell Script

2. run_general_tasks.sh

```
#!/bin/ksh
################################################################
################################################################
##
##  Filename: run_general_tasks.sh
##
##  Date of the last modification: 08/06/96
##
##  Description: Generic script called by /u/lynn/general_tasks.sh
##               to execute the task.
##
##  Arguments: <task_name>  : name of existing task
##             <flag     >  : -p or -h
##             <target   >  : either a profile manager name or hostname(s)
##
##
################################################################
################################################################
# Set the variable names

task_name=$1
flag=$2
target=$3


if [[ -z "$target" ]]; then
  echo "No TME Managed Nodes were selected."
  exit 1
fi

# execute the task against the profile manager or the hostnames passed
# from the calling script

for i in $target
  do
    wruntask $flag $i -l General_Tasks -t $task_name
  done

exit
```

Figure 487. Run_general_tasks Shell Script

You will notice that general_tasks.sh calls run_general_tasks.sh through the xnmappmon command. We would normally expect to run the following commands from general_tasks.sh:

```
for i in $target
  do
    xnmappmon -title "$task_name" -cmd  wruntask $flag $i \
    -l General_Tasks -t $task_name
  done
```

However, during our project we experienced some problems with xnmappmon passing the correct parameters to wruntask. We had to create run_general_tasks.sh to by-pass those problems.

3. Create a script file called netview_tasks.sh.

```
#!/bin/ksh
#####################################################################
#####################################################################
##
##  Filename: netview_tasks.sh
##
##  Date of the last modification: 08/06/96
##
##  Description: Korn shell script to run specific Netview task
##               Set_isTME_NODE against hosts or Collection selected
##               from the NetView map.
##
##  Arguments: < value > : True or False
##
##
#####################################################################
#####################################################################


value=$1

# set up Tivoli environment

 . /etc/Tivoli/setup_env.sh

# get the name of the Netview system running this script

NetviewHost=`echo $NVSessionHostname   cut -d "." -f1`

# Check if a Collection  rather than hosts was selected from the Map.
# Collections will always have a name ending in .CF
# $OVwSelections is the NetView variable that contains the name(s) of
# selected objects.

collection=`echo $OVwSelections   grep ".CF"`

# If this is not a Collection, pass the target hosts straight through.
# If it is a Collection, use the wtcoll program to list out the names of
# all members. Program "wtcoll" is documented in the ITSO Redbook
# Examples Using NetView for AIX Version 4, SG24-4515

if [[ -z "$collection" ]]; then
  target=$OVwSelections
else
  coll_name=`echo $OVwSelections   cut -d "." -f1`
  target=`/usr/local/bin/wtcoll -listSelectionNames $coll_name`
fi

xnmappmon -commandTitle "Set_isTME_NODE" \
          -cmd /usr/local/bin/run_netview_tasks.sh \
          $NetviewHost $value "$target"

exit
```

Figure 488. NetView_tasks.sh Shell Script

4. Create run_netview_tasks.sh.

```
#!/bin/ksh
################################################################
################################################################
##
##  Filename: run_netview_tasks.sh
##
##  Date of the last modification: 08/06/96
##
##  Description: Korn shell script called by netview_tasks.sh to
##               execute Set_isTME_NODE task.
##
##  Arguments: < NetviewHost > : name of Netview system
##             < value       > : True or False
##             < target      > : list of machines
##
##
################################################################
################################################################
NetviewHost=$1
value=$2
target=$3


if [[ -z "$target" ]]; then
  echo "No TME Managed Nodes were selected."
  exit 1
fi


for i in $target
  do
    wruntask -h  $NetviewHost -l Netview_Tasks -t Set_isTME_NODE \
             -a $i  -a isTME_NODE -a $value
  done

exit
```

Figure 489. Run_netview_tasks.sh Shell Script

The task Set_isTME_NODE is a little more complicated as it requires additional arguments, so it does not quite fit the generic script for Show_Filesystems and Who.

This task needs three parameters to be passed to it as well as the target on which it will run. The target must be the system running NetView for AIX.

- When the task is executed from the TME Desktop:

 Select the task endpoint from the available endpoints. You have to select a machine running NetView for AIX. Alternatively, the task could be set up as a TME job with the selected profile manager set to a profile manager that has subscribers of all NetView for AIX machines.

 We are prompted for the following fields:

 - Hostname to be changed
 - NetView for AIX field name, fixed at: isTME_NODE
 - True or False

- When we execute the task from NetView for AIX:

 The task endpoint is set from the NVSessionHostname variable.

 Alternatively, we could modify the code to execute against a profile manager that has subscribers of all NetView for AIX machines. The field isTME_NODE could then be changed on all systems running NetView for AIX.

The arguments required to run the task are set as follows:

- Hostnames to be changed are set from objects selected from the NetView for AIX map.
- NetView for AIX field name is fixed as isTME_NODE.

- However, the True/False field still needs to be set. This can be done in several ways:

 Define two separate menu items, one to set a value of True, the other to set a value of False. This is the method we used.

 In the script netview_tasks.sh, we could add some code to open an xterm which would execute a script to prompt for values:

```
tmpfile=/tmp/isTME_NODE.$$
xterm -geometry 50x5 -e /u/lynn/read_parms
value=`cat $tmpfile`
```

The script read_parms could do something as in Figure 490 on page 510.

```
#!/bin/ksh
print "Set isTME_NODE to True/False [True]: \c"
read value
if [[ -z "$type" ]]; then
  value=True
fi
echo "isTME_NODE will be set to $value"
echo "$value" > $tmpfile
sleep 3
exit
```

Figure 490. Code Excerpt to Prompt for Input in an Xterm

- We could use more sophisticated methods to enter the values such as creating SMIT panels. This would probably only be a reasonable solution if the requirement for additional parameters was reasonably complicated.

16.4 Creating NetView for AIX Collections from the Tivoli Database

In this section we show how to build a number of NetView for AIX collections from the information contained in the Tivoli object database. These examples extend the facility provided by the collection API program described in *Examples Using NetView for AIX V4*, SG24-4515.

In NetView, making collections of nodes allows you to treat those nodes as a single entity. It also automatically creates a submap containing all of the nodes in the collection. From this you can get an instant view of the status of a collection: which nodes are up and which are down. Collections are defined by *collection rules* , which may specify attributes of the node (all nodes within a network, or all nodes with particular entries in the NetView object database), or may just be a list. So for example, you could create a list of all of your key

servers in a collection. You would then automatically have a submap in which you could see the status of the key servers.

In the above example we suggested creating a logical group based on a list of nodes. That is exactly what you are doing in TME when you create managed nodes in a policy region, or subscribe them to a profile manager. The objective of this sample code is to provide a status view of these TME groupings by exposing them as collections in NetView, without having to define the list a second time.

This can be a useful facility, for example for checking the status of the managed nodes prior to performing TME 10 actions such as software distribution or task initiation.

16.4.1 Collection Definitions

The first example shows how to create TME collections for:

Policy regions Nodes contained in each of the policy regions.

Profile managers All nodes that subscribe to profile managers.

These collections are added to the NetView for AIX configuration dynamically using shell scripts and C programs. A number of NetView for AIX object fields are also created during this process.

The NetView for AIX server and TMR server do not need to be the same machine, but the NetView for AIX server must be a managed node and must have the TME roles to read the Tivoli database.

16.4.2 Creating the Policy Region Collections

The process we performed to create the collections is as follows:

- Start NetView for AIX.
- Add the new NetView for AIX object database fields needed by our example. There are two steps to this:
 1. Copy the new definitions (shown in Figure 491) into the fields subdirectory:

 `cp TME_fields /usr/OV/fields/C`

 2. Run `nv6000 -fields` to incorporate them in the database definition.

```
/*****
 ***
 * NetView for AIX Field definitions for the
 * Tivoli Collections
 ***
 *****/
 Field "isTME_NODE" {
      Type    Boolean;
      Flags   capability;
 }

Field "TME_Collection" {
      Type String;
      Flags Locate, General;
}
```

Figure 491. The TME_fields File

Next we run the sample script that creates the NetView for AIX collections:

- Execute script `create_tme_collections.sh` (see 16.4.2.1, “Sample to Place TME Node Groups in NetView Collections” on page 515).
- Check the tme_nv_coll.log file for any errors during execution.
- The NetView for AIX collections have now been defined and you will be able to see them in the collection editor dialog, as shown in Figure 492 and Figure 493.

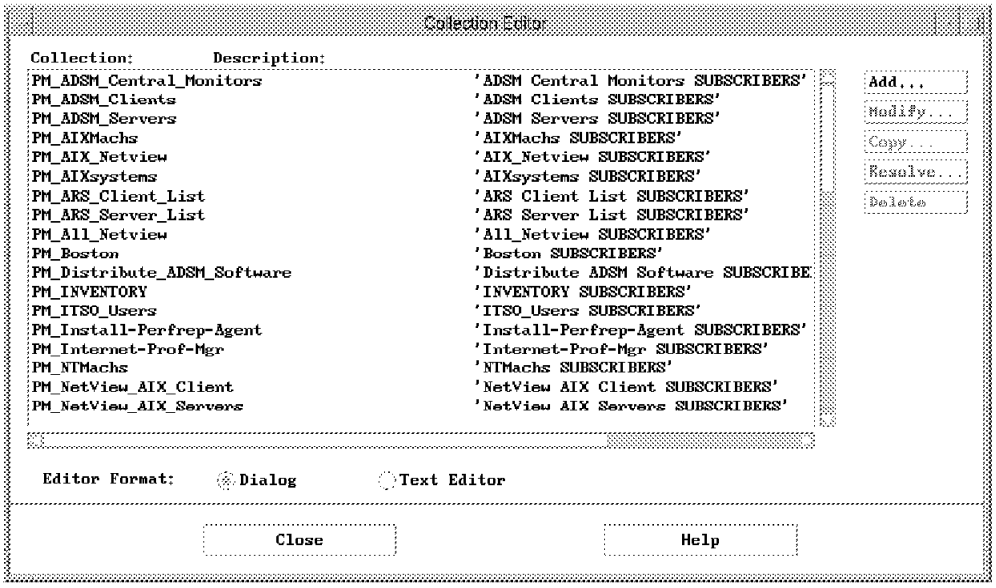


Figure 492. NetView for AIX Collections, Profile Managers

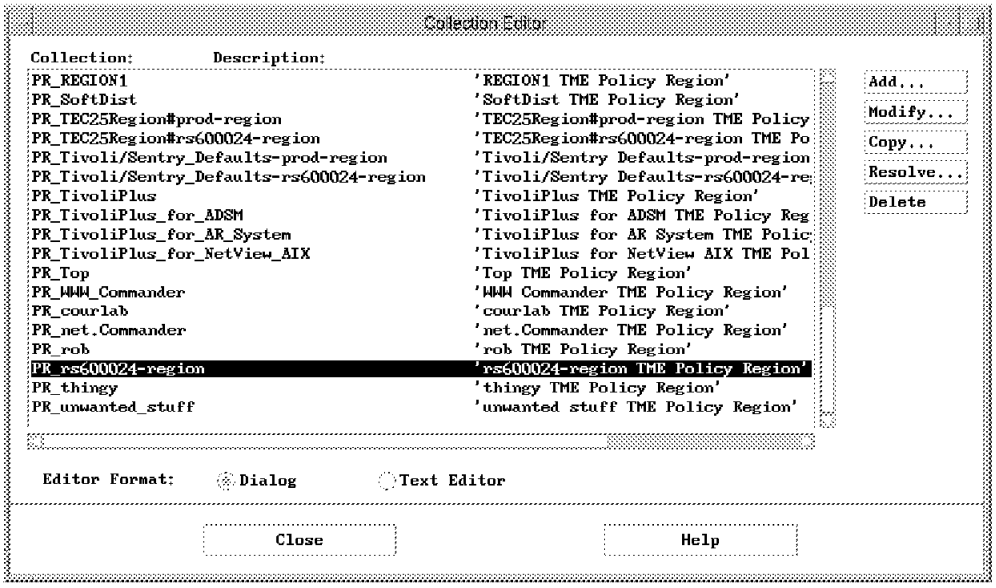


Figure 493. NetView for AIX Collections, Policy Regions

You can see thet there are two kinds of collection in the list. The collections whose names begin with PR_ contain the nodes created within specific policy regions. The ones whose names begin with PM_ contain the nodes that are subscribed to a specific profile manager. The collections are not simple lists,

but are rules that extract all nodes with a given value in the field in the NetView object database. This field was created and set by the create_tme_collections.sh shell script.

The profile manager collection definition differs from the Policy region subscriber definition because a new field is created in each node that is subscribed to a profile manager. This is because it is likely that one machine is subscribed to a number of different profile managers and will therefore be in a number of profile manager collections.

We illustrate the collection function with an example. Figure 494 shows the collection definition that the script created for collection PR_rs600024-region. This shows that the collection rule searches for all nodes which have the value rs600024-region in the object database field TME_Collection. This field was set by the shell script, based on information extracted from TME commands. Figure 495 on page 514 shows the result of resolving the collection rule, a list of all of the nodes in the policy region.

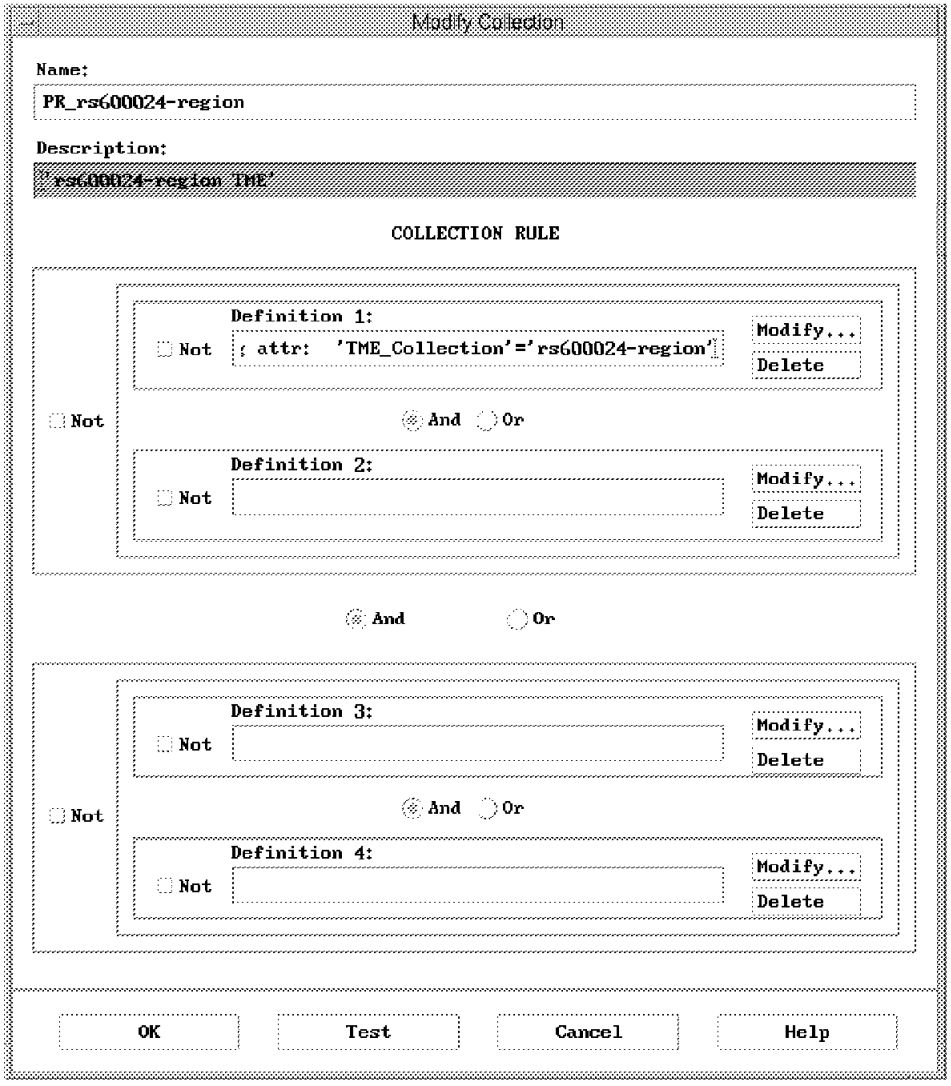


Figure 494. NetView for AIX Collections Definition

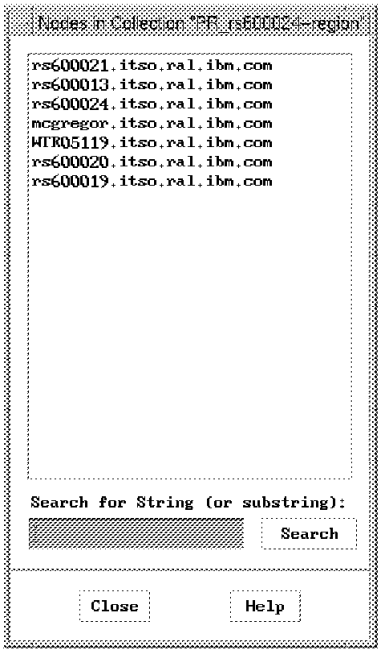


Figure 495. Resolved List of Nodes in Policy Region Collection

In addition to the policy region and profile manager collections, the script also creates a collection for all TME nodes. This is created in a similar way to the other collections, by setting the value of a field in the NetView object database. Figure 496 shows the NetView submap that represents these nodes. This submap shows status using color changes, using the standard NetView for AIX practice. Note that all of the collections will have submaps of this kind, each represented by an icon in the Collections submap and showing status in the same way.

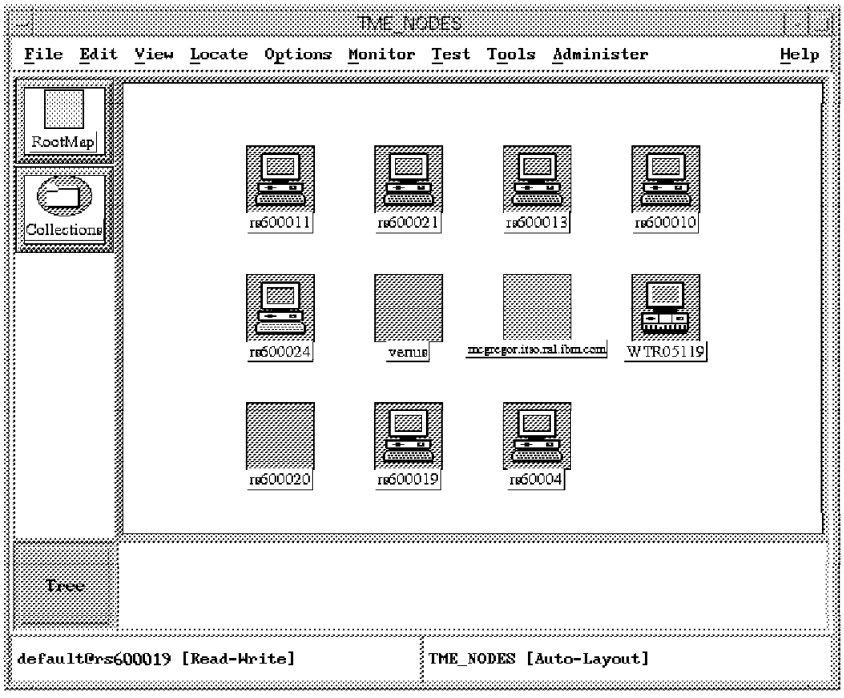


Figure 496. Collection Submap for All TME Nodes

16.4.2.1 Sample to Place TME Node Groups in NetView Collections

The example uses one main shell script create_tme_collections.sh to create all of the collections. That shell script, in turn, invokes two simple C programs. One of the programs is itsocoll, which gives command line access to a number of NetView collection API functions. This is the descendant of a previous program, called wtcoll, which is described in *Examples Using NetView for AIX Version 4*, SG24-4515. There is also a C program, set_tme_field, that sets a NetView object database field to a given value. create_tme_collections.sh is listed in Figure 497 and set_tme_field.c is listed in Figure 498 on page 517.

If you want to download these programs you can find them under the redbooks home page at http://www.redbooks.ibm.com.

```
#!/bin/ksh
#
# Script to create a number of collections
# from information contained in the Tivoli Database
#
# 1. All Managed Nodes
# 2. Policy Regions
# 3. Subscribers to Profile Managers
#
# Author: Paul Fearn IBM UK LTD.
#
ColHome=/:/u/paul/tme
CollLog=$ColHome/log/tme_nv_coll.log
CollTmp=/tmp/tme_nv.scratch

# Set TME Environment
. /etc/Tivoli/setup_env.sh

#####################################################
# Create Collection for All TME_NODES
#####################################################

ColRule="('isTME_NODE' = 'TRUE')"
ColName='TME_NODES'

$ColHome/wtcoll -CreateCol "$ColName" "$ColRule" "TME MANAGED NODE" 2>>$CollLog
#####################################################
#Policy Regions
#####################################################
echo "\nLookup policy Regions ... "

typeset -i i=0;
OIFS=$IFS

#Set IFS to equal a tab followed by a character return
IFS="
"
echo "Start Collection Creation process on `date`" >> $CollLog

POLICY_REGION=`wlookup -r PolicyRegion -a -L` >> $CollLog
for REGION in $POLICY_REGION
do
     echo "PR_$REGION ......."
     # Create A collection for each policy region
     # using TME_Collection NetView Variable (Defined prior running this script)

     # Define the NetView collection Rule
     ColRule="('TME_Collection' = '$REGION')"
     ColName=`echo $REGION   tr ' ' '_' `
     $ColHome/wtcoll -CreateCol "PR_$ColName" "$ColRule" "'$REGION TME
       Policy Region'" 2>>$CollLog
```

Figure 497 (Part 1 of 2). Shell Script create_tme_collections.sh

```
     # Set OVW Fields for each machine in a region
     for MACH_ID in `wls -l '@PolicyRegion:'$REGION 2>>$CollLog   grep
      ManagedNode   awk '{print $2}'`
     do
     SELECTION=`host $MACH_ID   awk '{print $1}'`
     if [ "$SELECTION" = "" ]
     then
          echo "Failed to resolve $REGION..May not a Managed Node" >>
               $CollLog
     else
             echo "Change Field TME_Collection for host $SELECTION "

             $ColHome/wtcoll -CreateField $SELECTION TME_Collection "$REGION"

             # Also set the isTME_NODE capability to True
           $ColHome/set_tme_field $SELECTION isTME_NODE "True"
        fi
     done
done
#####################################################
#Profile Managers
#####################################################

echo "\nLookup Profile Managers ... "

#Set IFS to equal a tab followed by a character return
IFS="
"

PROFILE_MANAGERS=`wlookup -r ProfileManager -a -L` 2>>$CollLog

for MANAGER in $PROFILE_MANAGERS
do
     echo "$MANAGER ......."
     i=0;

     # Create Relevant Field for each subscriber node

     for NODE in `wgetsub "@$MANAGER" 2>>$CollLog`
     do
          SUBSCRIBER=`nslookup $NODE   grep Name   awk '{print $2}' 2>>$CollLog`
          if [ "$SUBSCRIBER" != "" ]
          then
               echo "Subscriber is $SUBSCRIBER" >> $CollLog
                  echo "Node $NODE subscriber to $MANAGER"
               i=`expr $i+1`
               $ColHome/tme_nv_collection -CreateField $SUBSCRIBER "TME_SUB_$MANAGER"
                "Y" >> $CollLog
          else
               echo "$NODE not created; Maybe not a managed node" >> $CollLog
          fi
     done

     if [ $i -eq 0 ]
     then
          echo "No Subscribers to $MANAGER: Collection NOT Created"
     else
          # Define the NetView collection Rule
          ColRule="('TME_SUB_$MANAGER' = 'Y')"
          ColName=`echo $MANAGER   tr ' ' '_' `

          $ColHome/wtcoll -CreateCol "PM_$ColName" "$ColRule" "'$MANAGER SUBSCRIBERS'"
           2>>$CollLog
     fi
done

IFS=$OIFS

echo "Check $CollLog for any Errors"

exit 0
```

Figure 497 (Part 2 of 2). Shell Script create_tme_collections.sh

```
/* program to set a OVW database fields */
/* Arguments      */
/* $1 - Object Selection Name "eg rs60003.itso.ral.ibm.com"*/
/* $2 - OVW Database Field Name */
/* $3 - Value of Field */

#include <stdio.h>
#include <OV/ovw.h>
#include <OV/ovw_obj.h>

void usage()
{
  fprintf( stdout, "Usage:\n");
  fprintf( stdout, "set_tme_field <Selection Name> <OVW Database Field> <Value>");
  fprintf( stdout, "\n\n");
  fprintf( stdout, "Example: set_tme_field rs600024.itso.. isTME_NODE True");
  fprintf( stdout, "\n\n");
}

void main( int argc, char **argv)
{
char          *fieldName ;
OVwFieldId     fieldId ;
char           *selectionName ;
char           *value ;
int           var_boolean;
char           *collection ;
OVwObjectId      objectId ;

selectionName = argv[1] ;
fieldName = argv[2] ;
value = argv[3];

  if ( argc-- != 4 )
    {
      usage();
      exit(0);
    }

  var_boolean=99;

  if ( strcmp(value, "True") )
     var_boolean=0;
  else
     var_boolean=1;

  OVwDbInit() ;
  fieldId = OVwDbFieldNameToFieldId(fieldName) ;
  objectId = OVwDbSelectionNameToObjectId(selectionName) ;

     printf("object is %d\n", objectId);
     printf("field is %d\n", fieldId);


  if ( OVwDbSetFieldBooleanValue( objectId, fieldId, var_boolean ) == -1 )
    fprintf( stderr, "Error: Could not set %s to %s\n", fieldName, value);
  else
    fprintf( stderr, "Set %s Field value %s set to %s\n", selectionName , fieldName, value);

     exit();

}
```

Figure 498. Program set_tme_field.c

Appendix A. TME Configuration Repository

Tivoli/Inventory requires a relational database management system (RDBMS) to contain the configuration repository. The supported RDBMSs at the time of this project were Oracle 7.X and Sybase 10.X. For our project we choose Oracle 7.1.6.

During the Tivoli/Inventory installation, a script is run that defines the whole configuration repository scheme. This script can be found in the directory $BINDIR/TAS/RIM/SQL/scripts. The name of this script depends on the RDBMS used.

tivoli_ora_schema.sql For Oracle

tivoli_syb_schema.sql For Sybase

A.1 Example Data via the INVENTORYDATA View

The following example was taken after we initially scanned our environment. It shows all of the configuration data available through the INVENTORYDATA view, and since you have to provide values when setting up database queries, it will give you an idea of the values stored in the appropriate database fields.

```
**********************
BOOTED_OS_NAME
**********************

TME_OBJECT_LABEL                 BOOTED_OS_NAME
-------------------------------- ----------------
rs60008                          AIX
ntcli5s                          Windows 95
rs600012                         AIX
rs600011                         AIX
ntcli6s                          Windows
sun                              SunOS
rs60004                          AIX
hp                               HP-UX
rs600010                         AIX
venus                            AIX
ntcli7s                          Windows NT

**********************
BOOTED_OS_VERSION
**********************

TME_OBJECT_LABEL                 BOOTED_OS_VERSION
-------------------------------- ------------------
rs60008                          3.2
ntcli5s                          3.95
rs600012                         3.2
rs600011                         3.2
ntcli6s                          3.10
sun                              5.3
rs60004                          3.2
hp                               A.09.05.A
rs600010                         3.2
```

© Copyright IBM Corp. 1996

```
venus                           4.1
ntcli7s                         3.51

*********************
COMPUTER_ARCHITECTURE
*********************

TME_OBJECT_LABEL                COMPUTER_ARCHITECTURE
------------------------------- --------------------------
rs60008                         power
ntcli5s                         MCA
rs600012                        power
rs600011                        power
ntcli6s                         MCA
sun                             sun4m
rs60004                         power
hp
rs600010                        power
venus
ntcli7s                         PCI, ISA

*********************
COMPUTER_MODEL
*********************

TME_OBJECT_LABEL                COMPUTER_MODEL
------------------------------- --------------------------
rs60008
ntcli5s                         Unknown
rs600012
rs600011
ntcli6s                         IBM PS/2 Model 80
sun
rs60004
hp
rs600010
venus
ntcli7s                         Unknown

*********************
COMPUTER_SCANTIME
*********************

TME_OBJECT_LABEL                COMPUTER_SCANTIME
------------------------------- ------------------------------
rs60008                         1996-08-06 13:28:55
ntcli5s                         1996-08-05 11:32:26
rs600012                        1996-08-01 14:17:11
rs600011                        1996-08-02 18:39:45
ntcli6s                         1996-08-05 17:08:49
sun                             1996-08-08 11:54:31
rs60004                         1996-08-08 10:25:12
hp                              1996-08-08 11:51:53
rs600010                        1996-08-08 11:52:53
venus                           1996-08-08 11:54:31
ntcli7s                         1996-08-08 13:37:34
```

```
**********
HARDWARE_SYSTEM_ID
**********

TME_OBJECT_LABEL       HARDWARE_SYSTEM_ID
---------------------- ------------------------------------------------------
rs60008                2071979149.1.323#TMF_ManagedNode::Managed_Node#
ntcli5s                0974C+D8L6KCDXNLZ1PB00000521 [Tue Jul 23 23:28:33 1996]
rs600012               1525396064.8.7#TMF_ManagedNode::Managed_Node#
rs600011               1525396064.1.323#TMF_ManagedNode::Managed_Node#
ntcli6s                GOHLFDHTCYM5WZ1QB4J000000572 [Mon Jun 03 03:35:21 1996]
sun                    1525396064.18.7#TMF_ManagedNode::Managed_Node#
rs60004                2071979149.3.7#TMF_ManagedNode::Managed_Node#
hp                     1525396064.19.7#TMF_ManagedNode::Managed_Node#
rs600010               1525396064.12.7#TMF_ManagedNode::Managed_Node#
venus                  1525396064.15.7#TMF_ManagedNode::Managed_Node#
ntcli7s                F1ZMMJMBV44ORCWQ33VF00000571 [Tue Jul 30 22:08:15 1996]

**********
PAGING_SPACE_KB
**********

TME_OBJECT_LABEL                PAGING_SPACE_KB
------------------------------- ---------------
rs60008                                  147456
ntcli5s                                33042432
rs600012                                 147456
rs600011                                 278528
ntcli6s                                    1024
sun                                    177430528
rs60004                                   65536
hp                                     100654080
rs600010                                 262144
venus                                    131072
ntcli7s                                2147352576

**********
PHYSICAL_MEMORY_KB
**********

TME_OBJECT_LABEL                PHYSICAL_MEMORY_KB
------------------------------- ------------------
rs60008                                     131072
ntcli5s                                      32268
rs600012                                    131072
rs600011                                    262144
ntcli6s                                       8060
sun                                          65536
rs60004                                      32768
hp
rs600010                                    131072
venus                                        65536
ntcli7s                                      81332

**********
PROCESSOR_MODEL
**********
```

```
TME_OBJECT_LABEL                PROCESSOR_MODEL
------------------------------- -------------------------------
rs60008                         UNKNOWN
ntcli5s                         Intel Pentium
rs600012                        UNKNOWN
rs600011                        UNKNOWN
ntcli6s                         386 DX
sun                             sparc
rs60004                         UNKNOWN
hp                              PA-RISC_1.1
rs600010                        UNKNOWN
venus                           UNKNOWN
ntcli7s                         Intel Pentium

**********************
PROCESSOR_SPEED
**********************

TME_OBJECT_LABEL                PROCESSOR_SPEED
------------------------------- ---------------
rs60008
ntcli5s                                      66
rs600012
rs600011
ntcli6s                                      20
sun                                          50
rs60004
hp
rs600010
venus
ntcli7s                                     133
```

A.2 TME Configuration Repository Views

This section lists the default views and their data fields that are created when the configuration database is set up.

```
************** Begin of View ********************
BASENODE_VIEW
************************************************
 HARDWARE_SYSTEM_ID
 MANUFACTURER_ID
************** End of View **********************

************** Begin of View ********************
COMPUTER_VIEW
************************************************
 HARDWARE_SYSTEM_ID
 TME_OBJECT_LABEL
 COMPUTER_ARCHITECTURE
 COMPUTER_MODEL
 BOOTED_OS_NAME
 BOOTED_OS_VERSION
************** End of View **********************
```

```
************** Begin of View ******************
INSTALLED_SOFTWARE_VIEW
************************************************
 HARDWARE_SYSTEM_ID
 TME_OBJECT_ID
 SOFTWARE_ID
 SOFTWARE_VERSION_ID
************** End of View ********************

************** Begin of View ******************
INVENTORYDATA
************************************************
 HARDWARE_SYSTEM_ID
 TME_OBJECT_ID
 TME_OBJECT_LABEL
 COMPUTER_ARCHITECTURE
 COMPUTER_MODEL
 PHYSICAL_MEMORY_KB
 PAGING_SPACE_KB
 COMPUTER_SCANTIME
 PROCESSOR_MODEL
 PROCESSOR_SPEED
 BOOTED_OS_NAME
 BOOTED_OS_VERSION
 LOGICALDRIVE_NAME
 LOGICALDRIVE_MOUNTDIR
 LOGICALDRIVE_SIZE_KB
************** End of View ********************

************** Begin of View ******************
MANAGEDNODE_VIEW
************************************************
 HARDWARE_SYSTEM_ID
 TME_OBJECT_ID
 TME_OBJECT_LABEL
 COMPUTER_ARCHITECTURE
 BOOTED_OS_NAME
 BOOTED_OS_VERSION
************** End of View ********************

************** Begin of View ******************
MEMORY_VIEW
************************************************
 HARDWARE_SYSTEM_ID
 TME_OBJECT_ID
 PHYSICAL_MEMORY_KB
 PAGING_SPACE_KB
 COMPUTER_SCANTIME
************** End of View ********************

************** Begin of View ******************
PROCESSOR_VIEW
************************************************
 HARDWARE_SYSTEM_ID
 PROCESSOR_MODEL
 PROCESSOR_SPEED
************** End of View ********************
```

A.3 Configuration Repository Tables

This section lists the tables and their data fields that are created when the configuration database is set up.

```
************** Begin of Table *******************
ASSET
************************************************
 ASSET_ID
************** End of Table *********************

************** Begin of Table *******************
CDROM_DRIVE
************************************************
 CDROM_DRIVE_ID
 MANUFACTURER_ID
 CDROM_DRIVE_MODEL
************** End of Table *********************

************** Begin of Table *******************
COMMERCIAL_APPLICATION
************************************************
 SOFTWARE_ID
************** End of Table *********************

************** Begin of Table *******************
COMPONENT_MONITOR
************************************************
 MONITOR_NAME
 SOFTWARE_COMPONENT_NAME
 SOFTWARE_COMPONENT_VERSION
 COMPONENT_LANGUAGE_EDITION
 MONITOR_PROFILE_NAME
************** End of Table *********************

************** Begin of Table *******************
COMPONENT_MONITOR_PROFILE
************************************************
 SOFTWARE_COMPONENT_NAME
 SOFTWARE_COMPONENT_VERSION
 COMPONENT_LANGUAGE_EDITION
 MONITOR_PROFILE_NAME
************** End of Table *********************

************** Begin of Table *******************
COMPUTER_SYSTEM
************************************************
 HARDWARE_SYSTEM_ID
 COMPUTER_SCANTIME
 TME_OBJECT_ID
 TME_OBJECT_LABEL
 BOOTED_OS_NAME
 BOOTED_OS_VERSION
 COMPUTER_MODEL
 COMPUTER_ARCHITECTURE
 IS_A_REFERENCE_SYSTEM
************** End of Table *********************
```

```
************** Begin of Table ******************
COMPUTER_SYSTEM_MEMORY
************************************************
 HARDWARE_SYSTEM_ID
 PHYSICAL_MEMORY_KB
 PHYSICAL_MEMORY_DESC
 PAGING_SPACE_KB
 PAGING_SPACE_DESC
************** End of Table ********************

************** Begin of Table ******************
CONFIG_CHANGE_HISTORY
************************************************
 CONFIG_CHANGE_TYPE
 CONFIG_CHANGE_TIME
 CONFIG_CHANGE_DESC
************** End of Table ********************

************** Begin of Table ******************
CONTROLLER_CARD
************************************************
 DEVICE_CARD_ID
 CONTROLLER_CARD_MODEL
************** End of Table ********************

************** Begin of Table ******************
DESKTOP_PC
************************************************
 HARDWARE_SYSTEM_ID
************** End of Table ********************

************** Begin of Table ******************
DEVICE_CARD
************************************************
 DEVICE_CARD_ID
 MANUFACTURER_ID
************** End of Table ********************

************** Begin of Table ******************
DEVICE_DRIVER
************************************************
 SOFTWARE_ID
************** End of Table ********************

************** Begin of Table ******************
DEVICE_DRIVER_SOFTWARE
************************************************
 DEVICE_CARD_ID
 SOFTWARE_VERSION_ID
 SOFTWARE_ID
 VERSION_LANGUAGE_EDITION
************** End of Table ********************

************** Begin of Table ******************
DISK_FARM
************************************************
 HARDWARE_SYSTEM_ID
************** End of Table ********************
```

```
************** Begin of Table ********************
DOCKING_STATION
************************************************
 HARDWARE_SYSTEM_ID
************** End of Table **********************

************** Begin of Table ********************
FAXMODEM_CARD
************************************************
 DEVICE_CARD_ID
 FAXMODEM_CARD_MODEL
************** End of Table **********************

************** Begin of Table ********************
FLOPPYDRIVE
************************************************
 FLOPPYDRIVE_ID
 MANUFACTURER_ID
 FLOPPYDRIVE_TYPE
************** End of Table **********************

************** Begin of Table ********************
HARDDISK
************************************************
 HARDDISK_ID
 MANUFACTURER_ID
 HARDDISK_MODEL
 HARDDISK_SIZE_MB
 HARDDISK_ACCESS_SPEED
 HARDDISK_SEEK_TIME
 HARDDISK_HEADS
 HARDDISK_SECTORS
 HARDDISK_CYLINDERS
************** End of Table **********************

************** Begin of Table ********************
HARDWARE_SYSTEM
************************************************
 HARDWARE_SYSTEM_ID
 ASSET_ID
 LOCATION_ID
 MANUFACTURER_ID
 HARDWARE_SYSTEM_SERIAL_NUMBER
 HARDWARE_SYSTEM_DESCRIPTION
************** End of Table **********************

************** Begin of Table ********************
INHOUSE_SOFTWARE
************************************************
 SOFTWARE_ID
************** End of Table **********************

************** Begin of Table ********************
INSTALLED_CDROM_DRIVE
************************************************
 CDROM_DRIVE_ID
 INSTALLED_CD_ID
 CONFIG_CHANGE_TYPE
 HARDWARE_SYSTEM_ID
```

```
 CONFIG_CHANGE_TIME
************** End of Table *********************

************** Begin of Table *******************
INSTALLED_CONFIG_FILE
************************************************
 CONFIG_FILE_NAME
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TYPE
 CONFIG_FILE_PATH
 CONFIG_FILE_DATE
 CONFIG_FILE_TIME
 CONFIG_FILE_SIZE
 CONFIG_CHANGE_TIME
************** End of Table *********************

************** Begin of Table *******************
INSTALLED_COPROCESSOR
************************************************
 HARDWARE_SYSTEM_ID
 COPROC_MODEL
 COPROC_TYPE
************** End of Table *********************

************** Begin of Table *******************
INSTALLED_DEVICE_CARD
************************************************
 CONFIG_CHANGE_TYPE
 INSTALLED_CARD_INSTANCE
 DEVICE_CARD_ID
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TIME
************** End of Table *********************

************** Begin of Table *******************
INSTALLED_FLOPPYDRIVE
************************************************
 FLOPPYDRIVE_ID
 INSTALLED_FD_ID
 CONFIG_CHANGE_TYPE
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TIME
************** End of Table *********************

************** Begin of Table *******************
INSTALLED_HARDDISK
************************************************
 HARDDISK_ID
 INSTALLED_HD_ID
 CONFIG_CHANGE_TYPE
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TIME
 HARDDISK_SERIAL_NUMBER
************** End of Table *********************

************** Begin of Table *******************
INSTALLED_KEYBOARD
************************************************
 HARDWARE_SYSTEM_ID
```

```
 CONFIG_CHANGE_TYPE
 CONFIG_CHANGE_TIME
 MANUFACTURER_ID
 KEYBOARD_MODEL
************** End of Table **********************

************** Begin of Table ********************
INSTALLED_MONITOR
************************************************
 MONITOR_ID
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TYPE
 CONFIG_CHANGE_TIME
************** End of Table **********************

************** Begin of Table ********************
INSTALLED_MONITOR_PROFILE
************************************************
 HARDWARE_SYSTEM_ID
 SOFTWARE_COMPONENT_NAME
 SOFTWARE_COMPONENT_VERSION
 COMPONENT_LANGUAGE_EDITION
 MONITOR_PROFILE_NAME
************** End of Table **********************

************** Begin of Table ********************
INSTALLED_MOUSE
************************************************
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TYPE
 CONFIG_CHANGE_TIME
 MANUFACTURER_ID
 MOUSE_BUTTONS
 MOUSE_DRIVER_VERSION
 MOUSE_INTERRUPT_LINE
************** End of Table **********************

************** Begin of Table ********************
INSTALLED_PROCESSOR
************************************************
 PROCESSOR_NUMBER
 PROCESSOR_ID
 HARDWARE_SYSTEM_ID
************** End of Table **********************

************** Begin of Table ********************
INSTALLED_SIGNATURE_FILE
************************************************
 SIGNATURE_FILE_NAME
 SIGNATURE_FILE_SIZE
 SIGNATURE_FILE_PATH
 SIGNATURE_FILE_DATE
 SIGNATURE_FILE_TIME
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TYPE
 CONFIG_CHANGE_TIME
************** End of Table **********************
```

```
************** Begin of Table ******************
INSTALLED_SOFTWARE_VERSION
************************************************
 SOFTWARE_VERSION_ID
 SOFTWARE_ID
 HARDWARE_SYSTEM_ID
 VERSION_LANGUAGE_EDITION
 CONFIG_CHANGE_TYPE
 INSTALLED_VERSION_PATH
 CONFIG_CHANGE_TIME
************** End of Table ********************

************** Begin of Table ******************
INSTALLED_SW_COMPONENT
************************************************
 HARDWARE_SYSTEM_ID
 SOFTWARE_COMPONENT_NAME
 SOFTWARE_COMPONENT_VERSION
 COMPONENT_LANGUAGE_EDITION
 INSTALLED_FILEPACK_TIME
 INSTALLED_FILEPACK_PATH
 FILEPACK_ACTIVATED
 FILEPACK_ACTIVATION_TIME
 TME_ADMINISTRATOR_ID
************** End of Table ********************

************** Begin of Table ******************
INSTALLED_TAPE_DRIVE
************************************************
 TAPE_DRIVE_ID
 INSTALLED_TD_ID
 CONFIG_CHANGE_TYPE
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TIME
************** End of Table ********************

************** Begin of Table ******************
INSTALLED_UNKNOWN_FILE
************************************************
 HARDWARE_SYSTEM_ID
 INSTALLED_FILE_NAME
 INSTALLED_FILE_SIZE
 INSTALLED_FILE_PATH
 INSTALLED_FILE_DATE
 INSTALLED_FILE_TIME
 CONFIG_CHANGE_TYPE
 CONFIG_CHANGE_TIME
************** End of Table ********************

************** Begin of Table ******************
IO_CARD
************************************************
 DEVICE_CARD_ID
 IO_CARD_MODEL
************** End of Table ********************

************** Begin of Table ******************
LAT_LONG
************************************************
```

```
 LOCATION_ID
 LATITUDE
 LONGITUDE
************** End of Table **********************

************** Begin of Table ********************
LOCATION
************************************************
 LOCATION_ID
************** End of Table **********************

************** Begin of Table ********************
LOGICALDRIVE
************************************************
 CONFIG_CHANGE_TYPE
 LOGICALDRIVE_NAME
 HARDDISK_ID
 INSTALLED_HD_ID
 HARDWARE_SYSTEM_ID
 LOGICALDRIVE_SIZE_KB
 LOGICALDRIVE_FREE_KB
 LOGICALDRIVE_MOUNTDIR
 LOGICALDRIVE_SERIALNUMBER
 LOGICALDRIVE_VOLUMELABEL
 LOGICALDRIVE_FILESYSTEM
************** End of Table **********************

************** Begin of Table ********************
MANUFACTURER
************************************************
 MANUFACTURER_ID
 MANUFACTURER_NAME
************** End of Table **********************

************** Begin of Table ********************
MONITOR
************************************************
 MONITOR_ID
 MANUFACTURER_ID
 MONITOR_MODEL
 MONITOR_SIZE
************** End of Table **********************

************** Begin of Table ********************
NETWARE_SERVER
************************************************
 HARDWARE_SYSTEM_ID
 NW_DEVICE_NAME
 NW_VERSION
 NW_SUBVERSION
 NW_MAX_CONNS
 NW_MAX_VOLUMES
 NW_REVISION_LEVEL
 NW_SFT_LEVEL
 NW_TTS_LEVEL
 NW_MAX_CONNS_USED
 NW_ACCOUNTING_VER
 NW_VAP_VER
 NW_QUEING_VER
```

```
 NW_PRINTSERVER_VER
 NW_VIRTUAL_CONSOLE_VER
 NW_SECURITY_RESTRICT_LEVEL
 NW_INTERNET_BRIDGE_SUPPORT
 NW_CLIB_MAJOR_VER
 NW_CLIB_MINOR_VER
 NW_CLIB_REVISION
************** End of Table **********************

************** Begin of Table ******************
NETWORK_CARD
************************************************
 DEVICE_CARD_ID
 NETWORK_CARD_MODEL
************** End of Table **********************

************** Begin of Table ******************
NETWORK_NODE
************************************************
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TYPE
 NETWORK_NODE_ADDRESS
 CONFIG_CHANGE_TIME
 NETWORK_NODE_NAME
 NETWORK_PROTOCOL
************** End of Table **********************

************** Begin of Table ******************
NETWORK_SOFTWARE
************************************************
 SOFTWARE_ID
************** End of Table **********************

************** Begin of Table ******************
NT_INFO
************************************************
 HARDWARE_SYSTEM_ID
 NT_CURRENT_BUILD
 NT_CURRENT_TYPE
 NT_CURRENT_VERSION
 NT_REG_ORG
 NT_REG_OWNER
 NT_SYSTEM_ROOT
 NT_SERVICE_PACK
 NT_INSTALL_DATE
************** End of Table **********************

************** Begin of Table ******************
NW_VOLUMES
************************************************
 HARDWARE_SYSTEM_ID
 NWVOL_NAME
 NWVOL_TOTAL_BLOCKS
 NWVOL_BLOCK_SECTORS
 NWVOL_AVAILABLE_BLOCKS
 NWVOL_DIRECTORY_SLOTS
 NWVOL_AVAILABLE_SLOTS
 NWVOL_IS_REMOVABLE
************** End of Table **********************
```

```
************** Begin of Table ******************
OFFICE_BUILDING
************************************************
 LOCATION_ID
 BUILDING_ADDRESS
 BUILDING_DESCRIPTION
************** End of Table ********************

************** Begin of Table ******************
OPERATING_SYSTEM
************************************************
 SOFTWARE_ID
************** End of Table ********************

************** Begin of Table ******************
PC_BIOS
************************************************
 CONFIG_CHANGE_TYPE
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TIME
 PC_BIOS_ASSET_INFO
 PC_BIOS_SERVICE_TAG
 PC_BIOS_ID_BYTES
 PC_BIOS_DATE
************** End of Table ********************

************** Begin of Table ******************
PC_DEVICE_DRIVERS
************************************************
 CONFIG_CHANGE_TYPE
 PC_DEVICE_DRIVER_NAME
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TIME
 PC_DEVICE_DRIVER_START_ADDRESS
************** End of Table ********************

************** Begin of Table ******************
PC_IPX_LAN
************************************************
 CONFIG_CHANGE_TYPE
 IPX_ADDRESS
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TIME
 IPX_LOGIN_NAME
 IPX_FULL_NAME
************** End of Table ********************

************** Begin of Table ******************
PC_IPX_LAN_CONNECTIONS
************************************************
 CONFIG_CHANGE_TYPE
 PC_IPX_CONN_NAME
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TIME
 PC_IPX_CONN_NUMBER
************** End of Table ********************
```

```
************** Begin of Table ******************
PC_MEMORY
************************************************
 CONFIG_CHANGE_TYPE
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TIME
 CONVENTIONAL_TOTAL_KB
 EXTENDED_TOTAL_KB
 EXPANDED_TOTAL_KB
************** End of Table ********************

************** Begin of Table ******************
PC_PORTS
************************************************
 CONFIG_CHANGE_TYPE
 PC_PORT_BASE_ADDRESS
 HARDWARE_SYSTEM_ID
 CONFIG_CHANGE_TIME
 PC_PORT_TYPE
 PC_PORT_NUMBER
************** End of Table ********************

************** Begin of Table ******************
PERIPHERAL_SYSTEM
************************************************
 HARDWARE_SYSTEM_ID
************** End of Table ********************

************** Begin of Table ******************
PERSON
************************************************
 PERSON_ID
 PERSON_FIRSTNAME
 PERSON_LASTNAME
************** End of Table ********************

************** Begin of Table ******************
PERSON_LOCATION
************************************************
 PERSON_ID
 LOCATION_ID
************** End of Table ********************

************** Begin of Table ******************
PERSON_SYSTEM
************************************************
 HARDWARE_SYSTEM_ID
 PERSON_ID
************** End of Table ********************

************** Begin of Table ******************
PRINTER
************************************************
 HARDWARE_SYSTEM_ID
************** End of Table ********************

************** Begin of Table ******************
PROCESSOR
************************************************
```

```
 PROCESSOR_ID
 MANUFACTURER_ID
 PROCESSOR_MODEL
 PROCESSOR_SPEED
************** End of Table **********************

************** Begin of Table ********************
RDBMS_SOFTWARE
************************************************
 SOFTWARE_ID
************** End of Table **********************

************** Begin of Table ********************
REFERRED_SYSTEMS
************************************************
 HARDWARE_SYSTEM_ID
 REFERRED_SYSTEM_ID
************** End of Table **********************

************** Begin of Table ********************
ROOM
************************************************
 ROOM_FLOOR
 ROOM_NUMBER
 LOCATION_ID
 ROOM_DESCRIPTION
************** End of Table **********************

************** Begin of Table ********************
SOFTWARE
************************************************
 SOFTWARE_ID
 MANUFACTURER_ID
 SOFTWARE_NAME
 SOFTWARE_DESCRIPTION
************** End of Table **********************

************** Begin of Table ********************
SOFTWARE_COMPONENT
************************************************
 SOFTWARE_COMPONENT_NAME
 SOFTWARE_COMPONENT_VERSION
 COMPONENT_LANGUAGE_EDITION
 SW_FILEPACK_ID
 MANUFACTURER_ID
 COMPONENT_MAJOR_VERSION
 COMPONENT_MINOR_VERSION
 COMPONENT_REVISION
 SOFTWARE_COMPONENT_DESCRIPTION
 SOFTWARE_COMPONENT_PLATFORM_OS
 SOFTWARE_COMPONENT_PLATFORM_HW
 SOFTWARE_COMPONENT_FUNCTION
 SOFTWARE_COMPONENT_SERIAL_NUMB
 SOFTWARE_COMPONENT_ID_CODE
************** End of Table **********************

************** Begin of Table ********************
SOFTWARE_COMPONENT_FILE
************************************************
```

```
 SOFTWARE_COMPONENT_NAME
 SOFTWARE_COMPONENT_VERSION
 COMPONENT_LANGUAGE_EDITION
 SW_COMP_FILENAME
 SW_COMP_FILESIZE
************** End of Table **********************

************** Begin of Table ******************
SOFTWARE_COMPONENT_INSTANCE
************************************************
 SOFTWARE_VERSION_ID
 SOFTWARE_ID
 SOFTWARE_COMPONENT_NAME
 SOFTWARE_COMPONENT_VERSION
 VERSION_LANGUAGE_EDITION
 COMPONENT_LANGUAGE_EDITION
************** End of Table **********************

************** Begin of Table ******************
SOFTWARE_COMPONENT_TASK
************************************************
 SOFTWARE_COMPONENT_NAME
 SOFTWARE_COMPONENT_VERSION
 COMPONENT_LANGUAGE_EDITION
 TASK_NAME
************** End of Table **********************

************** Begin of Table ******************
SOFTWARE_FILEPACK
************************************************
 SW_FILEPACK_ID
 INSTALLATION_BEFORESCRIPT
 INSTALLATION_AFTERSCRIPT
************** End of Table **********************

************** Begin of Table ******************
SOFTWARE_SIGNATURE_FILE
************************************************
 SIGNATURE_FILE_NAME
 SIGNATURE_FILE_SIZE
 COMPONENT_LANGUAGE_EDITION
 SOFTWARE_COMPONENT_NAME
 SOFTWARE_COMPONENT_VERSION
 FILE_CHECKSUM
 FILE_CRC1
 FILE_CRC2
************** End of Table **********************

************** Begin of Table ******************
SOFTWARE_VERSION
************************************************
 SOFTWARE_VERSION_ID
 SOFTWARE_ID
 VERSION_LANGUAGE_EDITION
 ASSET_ID
 SOFTWARE_MAJOR_VERSION
 SOFTWARE_MINOR_VERSION
 SOFTWARE_REVISION
 SOFTWARE_BUILD
```

```
 SOFTWARE_VERSION_DESCRIPTION
 SOFTWARE_VERSION_SERIAL_NUMBER
 SOFTWARE_VERSION_ID_CODE
************** End of Table **********************

************** Begin of Table ********************
SOUND_CARD
************************************************
 DEVICE_CARD_ID
 SOUND_CARD_MODEL
************** End of Table **********************

************** Begin of Table ********************
SW_COMPONENT_DEPENDENCY
************************************************
 SOFTWARE_COMPONENT_NAME
 SOFTWARE_COMPONENT_VERSION
 COMPONENT_LANGUAGE_EDITION
 HARDWARE_SYSTEM_ID
************** End of Table **********************

************** Begin of Table ********************
SW_COMPONENT_RELATIONSHIP
************************************************
 COMPONENT_RELATIONSHIP_NAME
 COMPONENT_RELATED_COMPONENT
 SOFTWARE_COMPONENT_NAME
 SOFTWARE_COMPONENT_VERSION
 COMPONENT_LANGUAGE_EDITION
************** End of Table **********************

************** Begin of Table ********************
TAPE_DRIVE
************************************************
 TAPE_DRIVE_ID
 MANUFACTURER_ID
 TAPE_DRIVE_MODEL
************** End of Table **********************

************** Begin of Table ********************
UNIX_SERVER
************************************************
 HARDWARE_SYSTEM_ID
************** End of Table **********************

************** Begin of Table ********************
VIDEO_CARD
************************************************
 DEVICE_CARD_ID
 VIDEO_CARD_MODEL
 VIDEO_CARD_BIOS
 VESA_SUPPORT
 VESA_VERSION
 VESA_OEM_NAME
************** End of Table **********************
```

Appendix B. Special Notices

This publication is intended to help technical support personnel install and use TME systems management products for AIX. The information in this publication is not intended as the specification of any programming interfaces that are provided by TME 10 products. See the PUBLICATIONS section of the IBM Programming Announcement for Tivoli for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these

© Copyright IBM Corp. 1996

names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	IBM
NetView	MQ
OS/2	RS/6000
MQSeries	First Failure Support Technology
FFST	RS/6000
NetFinity	Trouble Ticket
SP	SystemView
PS/2	

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Java and HotJava are trademarks of Sun Microsystems Inc.

DCE	The Open Software Foundation
DynaText	Electronic Book Technologies, Incorporated
Excel	Microsoft Corporation
Hewlett-Packard	Hewlett-Packard Company
HP	Hewlett-Packard Company
HP/UX	Hewlett-Packard Company
Intel	Intel Corporation
IPX	Novell, Incorporated
Microsoft Excel	Microsoft Corporation
MS	Microsoft Corporation
NET	NCR Corporation
Netscape	logo Netscape Communications Corporation
NetWare	Novell, Incorporated
NFS	Sun Microsystems Incorporated
Novell	Novell, Incorporated
NT	Northern Telecom Limited or Microsoft Corporation
OpenView	Hewlett-Packard Company
Oracle 7	Oracle Corporation
ORACLE	Oracle Corporation
Pentium	Intel Corporation

PostScript	Adobe Systems, Incorporated
Sentry	American Telephone and Telegraph Company
SFT	Novell, Incorporated
Solaris	Sun Microsystems, Incorporated
Sun	Sun Microsystems, Incorporated
SunOS	Sun Microsystems, Incorporated
Sybase	Sybase Corporation
Tivoli Management Environment	Tivoli Systems Inc., an IBM Company
Tivoli Management Platform	Tivoli Systems Inc., an IBM Company
Tivoli Management Framework	Tivoli Systems Inc., an IBM Company
Tivoli	Tivoli Systems Inc., an IBM Company
Tivoli/Admin	Tivoli Systems Inc., an IBM Company
Tivoli/Courier	Tivoli Systems Inc., an IBM Company
Tivoli/Inventory	Tivoli Systems Inc., an IBM Company
Tivoli/Sentry	Tivoli Systems Inc., an IBM Company
Tivoli/Userlink	Tivoli Systems Inc., an IBM Company
Tivoli/Enterprise Console	Tivoli Systems Inc., an IBM Company
TivoliPlus	Tivoli Systems Inc., an IBM Company
Tivoli/AEF	Tivoli Systems Inc., an IBM Company
TME	Tivoli Systems Inc., an IBM Company
TME 10	Tivoli Systems Inc., an IBM Company
VESA	Video Electronics Standards Association
386	Intel Corporation

Other trademarks are trademarks of their respective companies.

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How To Get ITSO Redbooks” on page 543.

- *An Introduction to TME 10 Performance Management*, SG24-4644
- *Setting Up a TME 3.0 NT Environment*, SG24-4819
- *TME 3.0 NT - Automated Processes*, SG24-4793

C.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RISC System/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RISC System/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

C.3 Other Publications

These publications are also relevant as further information sources:

- *Tivoli Event Integration Facility User's Guide*, GC31-8337
- *Tivoli/Inventory User's Guide*, GC31-8381
- *Tivoli/Print User's Guide*, GC31-8392
- *Tivoli/Plus AR Systems*, GC31-8395
- *Tivoli/Plus for BoKs User's Guide*, GC31-8396
- *Tivoli/Plus for SEOS User's Guide*, GC31-8397
- *Tivoli/Plus for Pereguine User's Guide*, GC31-8399
- *Tivoli/Plus for Usison User's Guide*, GC31-8401
- *Tivoli/Plus User's Guide*, GC31-8402
- *TME 10 Tivoli/Plus ADSM User's Guide*, GC31-8405
- *TME 10 Tivoli/Plus NetWorker User's Guide*, GC31-8406
- *Tivoli/Courier Documentation Kit*, SK2T-6046

 This documentation kit contains:

© Copyright IBM Corp. 1996

 - *Tivoli/Courier User's Guide*
 - *Tivoli/Courier Reference Manual*
- *Tivoli Enterprise Console Documentation Kit*, SK2T-6050
- *Tivoli/Sentry Documentation Kit*, SK2T-6052
- *Tivoli/Administration Documentation Kit*, SK2T-6055
- *Tivoli/Management Platform Documentation Kit*, SK2T-6058

 This documentation kit contains:
 - *TME Desktop for Windows User's Guide*
 - *Tivoli Management Platform User's Guide*
 - *Tivoli Management Platform Planning and Installation Guide*
 - *Tivoli Management Platform Reference Guide*
- *Tivoli/ADE Documentation Kit Volume I*, SK2T-6062
- *Tivoli/ADE Documentation Kit Volume II*, SK2T-6063
- *Tivoli/ADE Documentation Kit Volume III*, SK2T-6064
- *Tivoli/ADE Documentation Kit Volume IV*, SK2T-6065

C.4 Other CD-ROMs

These CD-ROMs are also relevant as further information sources:

- *Tivoli TME 10 Software Distribution CD-ROM*, LCD4-0491
- *Tivoli/Courier Version 3.0 CD-ROM*, LK2T-6047
- *Tivoli/Inventory Version 3.0 CD-ROM*, LK2T-6049
- *Tivoli Enterprise Console/EIF Version 2.5D CD-ROM*, LK2T-6051
- *Tivoli/Sentry Version 3.0 CD-ROM*, LK2T-6053
- *Tivoli/Administration Version 3.0 CD-ROM*, LK2T-6056
- *Tivoli/Print CD-ROM*, LK2T-6057
- *Tivoli/Management Platform Version 3.1 CD-ROM*, LK2T-6059
- *Tivoli net.Commander CD-ROM*, LK2T-6060
- *Tivoli/Plus CD-ROM*, LK2T-6061
- *Tivoli/Management Platform CD-ROM*, LK2T-6066 (must be ordered with software product)

C.5 Other Diskettes

These diskettes are also relevant as further information sources:

- *TME 10 Tivoli/Administration PC-EndPoint Install Diskette*, LK2T-6067

How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

 To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

 To get lists of redbooks:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

 To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

 For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

 http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

 http://www.elink.ibmlink.ibm.com/pbl/pbl

 IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

 With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

© Copyright IBM Corp. 1996

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269	
Canada	1-403-267-4455	
Outside North America	(+45) 48 14 2207	(long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1) 415 855 43 29 (Outside USA)** — ask for:

 Index # 4421 Abstracts of new redbooks
 Index # 4422 IBM redbooks
 Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

 With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

List of Abbreviations

ACF	T/EC Adapter Configuration Facility
ADE	Tivoli/Application Development Environment
AEF	Tivoli/Application Extension Facility
AMS	Application Management Specification
BARC	before, after, removal, and commit configuration program(s)
BAROC	BAsic Recorder of Objects in C
BOA	Basic Object Adapter
CDF	component description file
CORBA	Common Object Request Broker Architecture
DES	Data Encrytion Standard
DHCP	Dynamic Host Configuration Protocol
DM	Distribution Manager (or Management)
DMI	Desktop Management Interface
DMTF	Desktop Management Task Force
EIF	Tivoli/Event Integration Facility
FFST	First Failure Support Technology
IBM	International Business Machines Corporation
IDL	Interface Definition Language
IMS	Internet Management Specification
ITSO	International Technical Support Organization
JMAPI	Java Management Application Programming Interface
MDist	multiplexed distribution
MIF	Management Information File
MLM	Systems Monitor Mid-Level Manager
NFS	Network File System
NIS	Network Information Services (formerly called ″NFS yellow pages″)
OID	object identifier
OMG	Object Management Group
ORB	object request broker
RCS	Revision Control System
RDBMS	relational database management system
RIM	RDBMS interface module
T/EC	Tivoli/Enterprise Console
TLL	Task Library Language
TME	Tivoli Management Environment
TMP	Tivoli Management Platform
TMR	Tivoli Management Region
TNWR	Tivoli NetWare repeater

© Copyright IBM Corp. 1996

Index

Special Characters

/etc/groups 254
/etc/hosts 254
/etc/inetd.conf 26
/etc/inittab 435
/etc/oratab 209
/etc/passwd 287
/etc/rc.nfs 26
/etc/security/passwd and /etc/security/user 287
/etc/services 26
/etc/Tivoli/setup_env.sh 45, 52
/tmp/client.cfg.error 51
/tmp/postemsg.out 449
/usr/local/Tivoli 23
/usr/local/Tivoli/docs/inv_user_guide.ps 211
/var/spool/Tivoli 23, 48
/var/spool/Tivoli/backups 53
.baroc files 356, 368, 372
.cds files 356, 368
 generating for logfile adapter 397
.conf files 356, 368
.err files 356
.fmt files 356, 379, 396
.oid files 356, 368, 414
@tivoli_ora_admin.sql 213
@tivoli_ora_schema.sql 214
$BINDIR 45
$DBDIR 45
$DISPLAY 44
$HOME/.profile 28
$LANG variable 436
$LIBDIR 45
\TIVOLI\SCAN 229
\TIVOLI\SCAN\INI 229
\TIVOLI\TMEAGENT\ directory 35

Numerics

0.default 456

A

abbreviations 547
accept 105
ACF 363
acronyms 547
action section in T/EC rules 384
action: keyword (T/EC rules) 390
actions, adding to AEF dialogs 493
activating a scanning process 224
Adapter Configuration Facility, see ACF
add a software reference model 242
add clients 30
add resources to administrator′s desktop 75
add scheduled job 66
ADE 455
administrator 7, 15, 72
 icon 72
 level 15
administrators
 defining for T/EC 324
 desktop 4
 for ACF 365
 for task libraries 455
 levels
 roles 8, 279
ADSM 428, 435
 client installation in /Plus module 436
AEF 455, 479
after distribution 124, 154, 178
aftmqm.bat 182
aftmqm.sh 157
AIX 3.2.5 system 157
AIX 4.1 system 157
Apache 442
APIs 455
append_log 128
Application Development Environment, see ADE
Application Extension Facility, see AEF
application management specification 106
architecture 3
assign reference model to system 244
asynchronous string interface (Sentry) 304
audit subsystem 452
authorization roles 6, 8, 19
automated tasks in T/EC 347, 402
available clients list 42

B

backup 52, 53
 authority roles 53
 directory 53
bandwidth 10
BAROC 329
 definitions for NetView/Openview 368
 extending 417
 definitions for Sentry 376
 description of BAROC 355
 example of creating new classes 394, 395
 message slots 337, 342, 393
BAT/EXE/COM Options using 178
befmqm.sh 157
before distribution 154
bi-directional communication 4
bibliography 541
bin 49

© Copyright IBM Corp. 1996

bootup 38
browse scheduled jobs 56
BufEvtPath definition 362

C

C preprocessor, see cpp
CD-ROM 31, 120, 208
CGI 449
change notification 245
character + 163
character ^ 116
class definition statement file, see .cds
CLI (command line interface)
 init.tecad_logfile 363
 postemsg 381
 rdsl 484
 waddaction 493
 waddegflt 340
 waddmon 376
 waddprop 480
 wassigneg 340
 wasync 304
 wcomprules 336, 391
 wcprb 335
 wcrtconsole 328, 340
 wcrteg 340
 wcrtjob 467
 wcrtpol 468
 wcrtprf 288, 365, 376
 wcrtprfmgr 288, 376, 467
 wcrtrb 335
 wcrttask 349
 wcrtusr 289
 wcrtusrcat 483
 wcrtusrsubcat 483
 wdelrbclass 418
 wdistrib 289, 376
 wgetpolm 288, 469
 wgettask 349
 wgetusr 289
 wimprbclass 336, 418
 wimprbrules 391
 winstall 428
 wloadrb 336, 391
 wlscurrb 335
 wlsdialog 484
 wlsinst 367
 wlspol 469, 476
 wlspolm 288, 469
 wlsrb 335, 354
 wlstlib 349
 wpopusrs 264, 288
 wpostemsg 381
 wputdialog 488
 wputpolm 288, 471
 wrcs 287
 wruntask 350, 409, 463
 wsetpr 254
CLI (command line interface) *(continued)*
 wsetusrs 289
 wstopesvr/wstartesvr 336
 wsub 289, 376, 467
 wtdbbackup 354
 wtdbclear 354
 wtdbspace 354
 wtdbstat 354
 wtdumprl 353, 410, 449
 wtll 455
CloseTivoliTicket 433
collection icon, Tivoli /Plus 427, 436
collection rules 510
commands, see CLI
commit 105, 144, 146, 191
 only 146
 operation 144, 191
commqm.bat 182, 197
compiling dialog files 485
compiling T/EC rules, see wcomprules
component description file (CDF) 106
compound reference models 244
COMPUTER_SYSTEM table 243
configuration program 105
configuration repository 231, 237, 241, 249
 configuration file for Net.Commander 447
 NetView and Openview 367
 NetView for AIX ruleset adapter 367, 370, 500
 NT logfile adapter 379
 scheme 241
 Sentry adapter 376
 event classes 376
 SNMP adapter 375
 the logfile adapter 356
console agent 34
control file data store 206
courier file package 104
cpp 459
create 72, 81, 82, 84, 85, 90, 91, 157, 174, 213, 219, 236, 237, 242
 administrator 85
 generic TME administrator 72
 inventory profile 219
 inventory user account on RDBMS 213
 jobs 84
 program option 157
 queries 237
 query libraries 236
 reference model 242
 staging area for source files 174
 task libraries 81, 90
 tasks 82, 91
crontab 67
current resources 29
customize inventory retrieval 221
customizing inventory profile 220

D

data encryption 20
database 212
 databases 207
 directory 49
 ID 212
 synchronization 61
db_setup.sh 209
DCE authentication 480
default
 default access method 30
 drive settings 36
 installation directory 36
default policies 220, 258, 467
 editing 259, 468
 generating default user IDs 272
 no default policy set 481
delete operation 63
deleting user IDs (Tivoli Admin) 264
deployment management 103
DES encryption 21, 27
desktop 39
Desktop Management Task Force 106
destination directory path 124, 154
destination drive 35
development environment 16
df 46, 91
dialog files, compiling
disk space free monitor 299
display on desktop 94
distribute 42, 105, 163, 188, 194
 all entries 163
 application load 42
 file package 105, 163, 188, 194
distribute to and distribute will 276, 285
 options 276, 285
distribution 163, 188, 194
 only 163
 options 163, 188
 type 163, 188, 194
distribution actions (Sentry) 307
drag and drop 64, 69, 97, 134, 160, 186, 478
dsl files 484
during commit 124, 178
DynaText browser 150

E

encryption 59
 keys 22
 level 59
 levels 27
 password 21, 59
endpoints 6, 15
error log 51
event 329, 340, 384
 rules 329
 section in T/EC rules 384
event *(continued)*
 server parameters 340
event adapters 319, 355
 configuration files 356, 362
 installation 355
event classes 329
 creating new classes 394
 for NetView/Openview 368
 hierarchy 355
 importing 332
event console 319
 configuring behavior of 343
 creation of 325
 icon 329
 using 341
event groups 336, 358
 CLI commands 340
event severity 341, 373
event sources 336, 358
event: keyword (T/EC rules) 390
EventServer resource 324, 360
EXACT COPY distribution option 283
exclamation point (!) 49
exec_task template (T/EC) 409
execute 93, 95
 tasks 93
execution privileges 92
export 106, 128
extend the INVENTORYDATA view 239

F

fan-out 107
FFST/6000 149
file browser 41, 50
file extensions 223
file package 103, 106, 118
 block (fpblock) 104
 definition 128
 distributing 160
 icon 160
 properties 120
FilePackage 116
filter definition (event adapters) 363
firewalls 451
First Failure Support Technology (FFST) 149
forward node (NetView rulesets) 374
frequency of updates 65

G

gadgets 488
gcoadd 51
generic executable 83
Get Program 125, 155, 156
get_diff_sw.sh 245
GID 285, 482
Group ID, see GID

group name 73

H

hardware inventory 231, 233
HARDWARE_SYSTEM table 243
history 140, 144, 167, 191
HP Openview
 event adapter 367, 413
HP/UX 435

I

icon bitmap format 337
import 106, 128
importing event classes 332, 369, 429, 448
indicator collections 310
INed 115
init.tecad_logfile 363
initial inventory scan 224
installation
 Configuration Repository Database 214
 Install Patch 40
 of event adapters 355
 of logfile adapter 357
 of NT logfile adapter 379
 of T/EC 321
 password 21, 30
 problems 44
 RDBMS 207
 Tivoli/Inventory 209
 updates 39
installer 208
installp 120, 132, 140, 157
 image 120
insufficient authorization 95
integration levels in TME 455
Intel LANDesk scanner 228
inter-region encryption 21
intermediate node 111
interval time 54
intra-region encryption 21
inventory 206, 217, 235
 database 235
 profile 217
 server 206
INVENTORYDATA 238, 241, 519
IP agent 31
IPX/SPX agent 31

J

jobs 67, 81, 464

K

Korn shell 45

L

LANG=C variable 28
LDAPPL.INI 228
LDISCAN 229
lib 49
license key 23, 428
list_inst_software.sh 245
list_ref_software.sh 245
log files 143, 148, 165, 171, 190
 for T/EC 354
log information options 122, 143, 148, 152, 165, 171, 176, 190
logfile adapter (T/EC) 356
 CDS file 397
 checking if installed 367
 configuration file 362
 updating using ACF 366
 configuring consoles for 358
 extending function 393
 extending to support a firewall system 451
 installation 357
 mapping messages to event slots 396
 starting and stopping 363
 troubleshooting 360, 398
 use by ADSM /Plus module 438, 439
logical operator 238
login 479
 AUTHENTICATION GRAMMAR 479
login name 72

M

mail servers (Net.Commander) 444
man files 49
managed nodes 6, 23, 118, 216, 231, 234
ManagedNode 29
management 19, 218, 253, 464
 by subscription 218, 253
 domain 19
memory requirements 79, 80
MERGEINI 229
message catalog facility 459
MIB objects in SNMP traps 414
MIF 205, 222, 224, 229
 file 222, 224, 229
mknod 305
MLM 413
monitor profiles (Sentry) 298
monitoring 294, 296, 303
 collections 294, 296
 schedule 303
move managed node 203
MQ Series event log example 393
mqm.sh 167, 172
MQSeries for AIX 149
MQSeries for NT 173, 174
msg_cat 49

multiplexed distribution (MDist) 107, 111

N

name 24, 60, 64, 67
 registry 60, 64, 67
 resolution 24
named pipe creation 305
namespace 15
NCSA 442
nested file package 104
Net.Commander 441
 creating a new web server 443
 description of 441
 prerequisites 441
Netscape 443
 Netscape Enterprise Server 443, 448
NetView DM 104, 106
 change file 104
 pre and post scripts 106
NetView for AIX 495
 collections 510
 configuring rulesets 373
 event adapter 367
 adding trap types 413
 mapping traps to baroc 418
 integration with T/EC 495
 menus, how to add TME functions 502
 registration files 503
 ruleset adapter 370, 420
 baroc definition 372
 rulesets 497
NetWare 271
network 10, 107
 bandwidth tuning 107
 topology 10
NFS 50
NNTP servers (Net.Commander) 450
nobody user ID 449
non-zero exit code 155
notice 9, 165, 196, 202
 group 9, 165, 196, 202
 messages 165
NT, see Windows NT
nvevents 495
nvserverd 332, 370
nvserverd.baroc 372, 390, 422

O

odadmin 48, 59, 112
one-way 5, 11, 59
 connection 5, 11, 59
 interconnected 11
ORACLE 207, 208
 installation script 208
oserv 24, 48, 51, 52
 database 51
 oservlog 52
oserv *(continued)*
 shutdown 24
overall RAM 79

P

PageSp/totalfree 77
paging space 79
panel layout options 483
parallel 84, 107
 distribution 107
patch 52
PATCHES.LST 40
PC 6, 23, 31, 118, 206, 214, 223, 234, 242
 configuration file 223
 managed node 6, 23, 31, 118, 214, 234, 242
 scanning 206, 214
 inventory 206
performance 76
 considerations 76
 toolkit 76
planning Sentry monitors 314
platform-specific 113, 123
 options 123
policies
 default and validation 288
 for user ID profiles 258
 modifying default policy 455
policy objects 467
 to create 468
policy regions 5, 12, 18, 61, 62, 70, 110, 236
 adding Tivoli/Admin resources to 254
 changing policy objects 471, 474
 creating 61
 creating NetView collections from 511
 deciding how to organize 296
 for Net.Commander 442
 icon 70, 236
 moving objects between 473
populate profile managers of file packages 115
postemsg 381, 449
Postscript documentation 210
preinstallation script 25
product install 40, 43
 dialog 43
production environment 16
profile 6, 7, 14, 68, 69, 70, 103, 118
 distribution 14
 endpoint 69
 subscribing 69
 manager 7, 68, 69, 70, 118
 defining 69
 icon 70
 subscribing 69
profile managers
 as task library targets 464
 creating NetView collections from 512
 for Tivoli/Admin 255, 268
 for Tivoli/Sentry 296, 297

profile managers *(continued)*
hierarchy of 269
used by ADSM /Plus module 438
used by Net.Commander 444
used by Remedy /Plus module 430, 432
program 124, 125, 154, 155, 156
name 125, 155, 156
options 124, 154
properties 234
property 237, 238
name 237
value 238
PTF U443133 371, 389, 497
ptxtab 78
pull 65, 105
push 65, 105

Q

Query push button 70, 250

R

RCS, see Revision Control System
rdsl 484
Real/comp 77
Real/noncomp 77
reboot 180, 191
machine 180
reference model 242
registration file for NetView event display 495
reinstall 24, 49
Remedy Action Request System 428
integration with T/EC 432
remmqm.bat 182
remmqm.sh 157
remote connection 59
remote site 5
management 5
operation 5
removability 105
remove operation 63, 105, 140, 167
repeat the job 68
repeater 107, 111, 160
site 111
resolution of links 124, 154
resource 5, 62
types 62
restrictions field 55
retry options field 55
return code 164
revision control system 287
roles 15, 92
required to execute task 92
roles, see administrators - roles
root 7, 70
root user ID 264
rule builder (T/EC) 384
rulebase for T/EC 329, 384
CLI commands 335
copying 331
default 330
loading 331
Ruleset Editor 497
run queries against other views 240

S

scanning 205
script output 172
secure connection 59
security 20
select media 41, 210
select product 41
sentry engine 293
Sentry, see TME 10 distributed monitoring 295
Sentry.baroc file 376
serial 84
server database 22, 47
server ID 212
ServerLocation definition 362
service 34, 203
agent 34
pack 203
set 73, 74, 75, 86, 87, 88
logins 74, 87
notice groups 75, 88
resource roles 73, 87
TMR roles 73, 86
setup program 33
shell service 60
signatures 205
simple encryption 20, 27
level 20
simulated message catalog 459
sizing 76
slots in BAROC messages 337, 342, 393
slow network connection 104
SNMP adapter 375
SNMP OID mapping file 368, 414
software 223, 230, 232, 249
distribution 249
queries 249
inventory 232
signatures 223, 230
software distribution of MQSeries for NT 185
source directories & files 122, 152, 176
source host 104, 122, 125, 152, 176
space problems 45
specific trap specification 390
SQL statement 237
sqlplus 214
staged 84
startup 36, 39
group 39
options 36

status information 43
status of scanning 225
subregions 6, 18, 61
subscriber icons 115
subscribers 14, 114
 for Tivoli Admin 267, 273
SysInfo 231
syslog configuration (Sentry monitor) 304
syslog subsystem 452
syslog_filter.ksh 306

T

T/EC 319
 adapter configuration facility (ACF) 363
 components of 319
 configuration 323
 creating consoles 325
 debugging commands 353, 360, 410
 defining administrators 324
 event 336, 341, 358, 383
 importing 391
 processing rules 383
 severity 341
 sources 336, 358
 tracing 391
 event adapters 355
 configuration file 362
 event classes 329, 355
 creating new classes 394
 importing 332
 event groups 336, 358
 generating events from the command line 381
 installation 321
 integrating with NetView for AIX 495
 integration with ADSM /Plus module 439
 integration with Net.Commander 447
 logfile adapter 357
 debugging 398
 mapping messages to event classes 396
 NetView adapter 367, 413
 NetView ruleset adapter 370
 Openview adapter 367, 413
 planning 320
 rule builder GUI 384
 rulebase 329, 357
 CLI commands 335
 copying 331
 loading 331
 Sentry adapter 376
 server parameters 340
 SNMP adapter 375
 tasks 344
 automatic task example 402
 creating new tasks 349
 trouble ticket integration 428
 Windows NT logfile adapter 379
T/EC server 80
task 67, 81
task libraries 67, 81, 82
 creating jobs from tasks
 exporting task definitions 455
 how to define tasks using TLL 455
 icon 82
 passing arguments to tasks 457, 461
Task Library Language, see TLL 99
TaskLibrary 81
Tasks for T/EC 344, 347
 CLI commands 349
 creating new tasks 349
tecad_logfile.baroc 439
tecad_logfile.cds file 398, 436, 439
tecad_logfile.fmt file 396, 436, 452
tecad_nv6k.baroc file 417
tecad_nv6k.cds file 418
tecad_nv6k.oid file 414
temporary area 105
test environment 16
timeout value 84
Tivoli 28, 70, 486
 administrator 70
 preview 486
Tivoli Management Environment 5
Tivoli Management Platform 4
Tivoli Management Regions 5
Tivoli Sentry, see TME 10 Distributed Monitoring 293
tivoli_ora_schema.sql 519
tivoli_syb_schema.sql 519
Tivoli/Admin, see TME10 User Admin 253
Tivoli/Plus Modules 427
 ADSM 435
 prerequisites 435
 use of Courier file packages 435, 437
 use of T/EC logfile adapter 435
 collection icon 427
 definition of 427
 remedy action request system 428
Tivoli/UserLink 105, 116
TIVSCAN 228
tl_def_prof_mgrs 469
TLL 455, 457
 compiling definitions 459
TME 31, 40, 50, 67, 107, 206, 241
 authorization roles 107
 client installation problems 50
 clients 31
 configuration repository 206, 241
 scheduler 67
 service pack installation 40
TME 10 Distributed Monitoring 293
 asynchronous string interface 304
 creating profiles 298
 distribution actions 307
 example of CLI commands 376
 indicator collections 310
 monitor actions 299

TME 10 Distributed Monitoring *(continued)*
monitoring collections
monitoring schedule
planning 314
profile managers for 296
sentry monitors in ADSM /Plus module 435
sentry monitors in Net.Commander 446
sentry monitors in Remedy /Plus module 428
T/EC event creation 376
TME 10 Enterprise Console, see T/EC
TME 10 NetView, see NetView for AIX 495
TME 10 User Admin 253
categories 483
creating subcategories 483
creating user ID profiles 255, 271
defining subscribers 267
deleting users 264, 286
distributing user profiles 267, 276
editing user profiles 270, 285
errors 262
EXACT COPY distribution 283, 285
extending function of 479
generating defaults 272
populating user profiles 261
profile policies 258
example of 260
profile types 254, 256
user properties 479
TME query facility 235
TMR connection 59
TMR database synchronization 65
TMR sizing 10
top level policy regions 61, 63
tracing T/EC rules 391
trouble ticket 428
troubleticket.sh script 432
trusted host 30, 60
facility 60
two-way 5, 11, 18, 110
connections 5, 18, 110
interconnected 11

U

UID 260, 264, 285, 482
multiple definitions 276
unattended installation 174
universal monitoring collection 301
UNIX 206, 231, 482
inventory scanning 206
login 482
scanning 231
program 231
user ID, see UID
updates 65, 66
manual 65
scheduled 65
upon removal 124, 142, 154, 170, 178
user ID management 254, 261
multiple system types 271
user ID profiles 479
user profiles - creating 255
distributing 268, 276
example of 275
populating 261
user profiles - modifying 285
user-written query 233
user_name@hostname 75

V

validation policies 258, 467, 475
errors 262, 478

W

waddaction 493
waddegflt 340
waddmon 376
waddprop 480, 489
wassigneg 340
wasync 304
wchkdb 51, 52
wchkdb -u 52
wcomprules 336, 369, 387, 391
wcpfpblock 104
wcprb 335
wcrtadmin 76, 92
wcrtconsole 328, 340
wcrteg 340
wcrtfpblock 104
wcrtjob 467
wcrtpol 468
wcrtprf 288, 376
wcrtprfmgr 288, 376, 467
wcrtrb 335
wcrttask 93, 349
wcrttlib 93
wcrtusr 289
wcrtusrcat 483
wcrtusrsubcat 483
wdelrbclass 418
wdistfpblock 104
wdistrib 289, 376
Web server - monitoring example 295
wgetpolm 288, 469
wgettask 349
wgetusr 289
when job complete 68
where keyword (T/EC rules) 390
wimprbclass 336, 369, 418
wimprbrules 391
Windows NT 271
event adapter 379
winstall 428
wlistlib 349

wloadrb 336, 369, 391
wlookup 51
wlookup -R 52
wlookup -r <resourcetype> -a command 52
wlscurrb 335
wlsdialog 484
wlsinst 367
wlspol 469, 476
wlspolm 288, 469
wlsrb 335, 354
wpopusrs 264, 288
wpostemsg 360, 381
wpreinst.sh 25
wputdialog 488
wputpolm 288, 471
wrcs 287
wrmfpblock 104
wrpt 107, 112
wruntask 98, 350, 409, 463
wserver 44
wsetpr 254
wsetrimpw 214
wsetusrs 289
wstopesvr/wstartesvr 336
wsub 289, 376, 467
wtcoll 515
wtdbbackup 354
wtdbclear 354
wtdbspace 354
wtdbstat 354
wtdumprl 353, 410, 449
wtll 455

X

X-windows 45
X11 50
xhost 45

IBM®

Printed in U.S.A.

